Vom Fachbereich für Mathematik und Informatik
der Technischen Universität Braunschweig
**genehmigte Dissertation**
zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)

**Thomas Lindner**

# Train Schedule Optimization in Public Rail Transport

**30. Juni 2000**

1. Referent: Prof. Dr. Uwe T. Zimmermann
2. Referent: Priv.-Doz. Dr. Michael L. Dowling
eingereicht am: 19. April 2000

# Acknowledgements

Many people have contributed to this thesis in one way or the other. I would like to thank Uwe Zimmermann and the other members of the Department of Mathematical Optimization for their support. The motivating atmosphere in the department provided an excellent framework for scientific working.

I would also like to express my gratitude to Leo Kroon, Alexander Schrijver and Matthias Krista for making test data and other railroad-related information available. It has been very helpful to know Robert Bixby. Without him, I would never have found the 'hidden parameters' in the CPLEX software which accelerated the MIP solution process by a factor of 10 in some cases.

Special thanks go to Karl Nachtigall. It has been a pleasure working together with him on chapter 3. I hope that we will be able to solve the PESP instance 14 some day. This instance is officially known as 'Jul18', because it was generated July 18 some years ago and nobody can remember the exact year!

My last special thanks go to Michael Bussieck, not only for those many scientific discussions, but also for personal support and – probably the most important – for encouraging me to start writing a thesis on train scheduling.

# Preface

This thesis deals with *train scheduling problems* with an emphasis on *public rail transport.* In particular, we assume a *periodic schedule* and a fixed railroad track *network*, which is common for public rail transport.

The fundamental mathematical model discussed here is the *Periodic Event Scheduling Problem (PESP)* introduced by Serafini and Ukovich in 1989. In a few words, the PESP is the problem of finding a feasible schedule for some periodically recurring events subject to certain constraints. The PESP is known to be NP-complete and therefore belongs to a class of problems assumed to be *very hard*.

We will analyze different existing algorithms for solving PESP instances. Based on this investigations, we modify these algorithms to achieve a much better performance for problem instances from practice. Furthermore, we discuss *polyhedral* aspects of a *mixed integer programming (MIP)* formulation of the PESP, thereby deriving *valid inequalities* and proving some properties of these inequalities. We combine existing algorithmic ideas with new ideas from these polyhedral investigations in order to obtain a new algorithm that can be successfully applied to PESP instances.

There are many criteria for evaluating schedules. The PESP itself is a feasibility problem. We extend it by an objective function representing the *operational costs* of realizing a schedule. The cost approach is based on a model suggested by Claessens. The resulting model is called *minimum cost scheduling problem (MCSP)*.

The decision version of the MCSP is shown to be NP-complete. We present a MIP formulation of the problem. With the help of *polyhedral* methods like *preprocessing* techniques, *valid inequalities*, a specific *relaxation*, a *branch-and-bound* and a *cutting plane* procedure, we are able to solve real-world instances of the MCSP, which is not possible within a reasonable amount of time when using the direct MIP formulation and a commercial MIP solver.

The mathematical models and algorithms introduced in this thesis are tested on practical instances obtained from the railroad companies of Germany (*Deutsche Bahn AG*) and the Netherlands (*Nederlandse Spoorwegen*).

The cost approach of the MCTP belongs to the *strategic planning* methods, i.e. it is used to evaluate possible scenarios 5–15 years ahead in the future. Our experiences show that it is possible to produce solutions of the MCSP for practically relevant problem sizes in a few minutes, which is acceptable for strategic planning. Moreover, our algorithm determines *lower bounds* on the costs and thus enables us to give bounds on the *quality* of the solutions (if we are not able to solve the problem instance *exactly*).

An important point is the transfer of mathematical models / ideas into practice. Mathematical ideas tend to be *abstract* and *non-intuitive* and are therefore disregarded by practitioners if they are not carefully introduced. In order to overcome these obstacles, the *German Federal Ministry of Education and Research* funded a series of projects on *Mathematical Methods for Solving Problems in Industry and Business*. In these joint projects, mathematicians, engineers and software developers work together, transferring mathematical ideas into practical software. Application fields are, for example, traffic, logistics, medicine or finance. This thesis emerged from the project *Train Schedule Optimization in Public Transportation.*

The thesis is organized as follows: In chapter 1, we give an introduction to traffic planning in general and with respect to schedules. Chapter 2 provides an overview of existing models for train scheduling and includes some extensions of the models. In particular, the PESP and the MCSP are described. Furthermore, we discuss computational complexity aspects of the PESP and the MCSP. In chapter 3, the PESP is investigated in detail. We present existing algorithms for solving PESP instances and develop modifications and new algorithms that allow a much faster solution of such instances. We also give theoretical results on the polyhedral structure of the PESP. In chapter 4, we introduce algorithms for solving MCSP instances. With the help of a decomposition idea, we develop a relaxation iteration and a branch-and-bound approach for the MCSP. Both methods require the solution of certain subproblems, which are also examined. Chapter 5 contains computational results for our real-world test instances, and the last chapter deals with conclusions and suggestions for further research.

# Contents

# Chapter 1

# Public Rail Transport Planning

Nowadays, public rail transport planning is a highly complex task. Too many objects interact with each other to be manageable simultaneously (cf. table 1.1, details are found in the *Geschäftsbericht der Deutschen Bahn* [20]). Various subproblems of different nature like network design, scheduling or routing occur, and the solutions of most of those subproblems depend on the solutions of the other subproblems. Due to severe competition from other transportation modes, the rail industry is eager to improve its operational efficiency and rationalize its planning decisions. Analytical models get more and more important in supporting managerial decision-making. The process of privatization of public transportation companies enforces the efficient utilization of resources.

| | |
|---|---|
| 38000 | km of network |
| 40000 | trains per day |
| 66 billion | traveler kilometers |
| 15 billion | gross investment (in DM) |
| 250000 | employees |
| 130 | license agreements with other railroad companies |

Table 1.1: Reference numbers of the German railroad company *Deutsche Bahn AG* (1998)

Different demands on the transport service come from the different departments of a railroad company. The *marketing* departments request taking care of the passengers' wishes like minimization of travel time, pleasant changes from one train to another (short waiting time, opposite platforms). The logistic departments pay attention to the *cost* aspects. They are responsible for the efficient usage of rolling material and personnel. Available rolling stock has to be considered as well as crew rulings. The departments maintaining the network take care of *operational constraints* occurring, for example, for concurrent use of critical points (single tracks, stations, switches, signals). All these, usually conflicting, demands are shown in figure 1.1.

Apart from economical aspects, political decisions and prestigious investment projects influence the planning process. In 1995, Baron [3] describes the situation of public transportation in Germany, con-

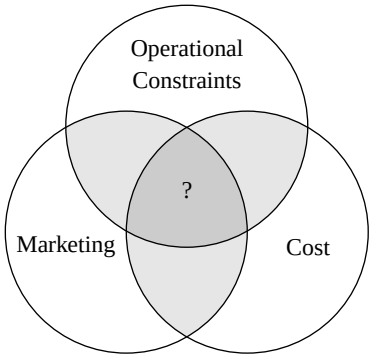



Figure 1.1: Conflicting demands

cluding that *transport policy and planning will remain a playing field of scientists, lobbyists, politicians, gurus, fanatics and concerned citizens for many years to come, and it will keep generations of journalists busy.*

Due to the tremendous size of the public rail traffic system, the planning process is divided into several steps (also see [10]). A diagram of this hierarchical decomposition is given in figure 1.2.

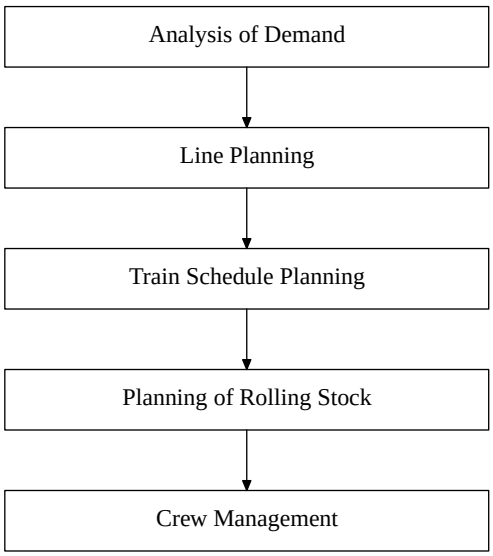



Figure 1.2: Hierarchical planning process

In a first step, the *passenger demand* has to be analyzed. As a result, the amount of travelers wishing to go from certain origins to certain destinations is known. As a subsequent task, *lines* are determined, i.e. routes where trains run. Also, the frequencies for the lines are determined. Afterwards, in the *train schedule planning* step, all arrival and departure times of the lines are fixed. This has to be done subject to the *periodicity* of the system (the German railroad train schedule operates with a period of one hour, for example). Now engines and coaches have to be assembled to trains, which are assigned to lines. This is called *planning of rolling stock*. A similar task is the *crew management*, which means

the distribution of personnel in order to guarantee that each train is equipped with the necessary staff.

Every single step in this process is a difficult task. We will discuss these steps further in section 1.1. A problem of the decomposition is that the optimal solution for one part serves as fixed input for the subsequent problem. One cannot expect an overall optimal solution in the end. It is even possible that at some point, former decisions have to be changed, and a part or the complete process has to be repeated. Nevertheless, this hierarchy provides a partition of the traffic planning problem into manageable tasks.

Another classical point of view [2, 33] is the partition of the planning process into *strategic*, *tactical* and *operational* planning level, table 1.2.

| Planning level | Time horizon | Goal |
|---|---|---|
| Strategic | 5–15 years | Resource acquisition |
| Tactical | 1–5 years | Resource allocation |
| Operational | 24 h – 1 year | Day-by-day decisions |

Table 1.2: Planning levels

On the strategic planning level, possible infrastructure investments are examined. The goal is to decide about resource acquisition (i.e. building new traffic links etc.). Such projects may have a duration of 5–15 years, and thus the view of the future plays an important role. The analysis of passenger demand and the design of line plans belong to this planning level. It is also possible to examine train schedules at this point of time, e.g. in order to examine the effect of a certain infrastructure proposal on the travel time.

The tactical planning level focuses on resource allocation in the medium term. Here, the general pattern of traffic flow is derived from invariable infrastructure and customer demand data. More detailed line plans and train schedules are developed, as well as general patterns for rolling stock circulation and crews.

Day-by-day decisions constitute the operational planning level. Here, due to unexpected events like breakdowns, special trains or short term changes in the infrastructure caused by construction sites, parts of a schedule or rolling stock and crew assignment patterns have to be rearranged.

During the last decade, the use of mathematical optimization models for rail transport planning and thus the automatic computation of line plans, schedules, crew patterns etc. has increased significantly (for an overview we refer to [18]). In the eighties, the application and development of mathematical models was hindered by insufficient computational capabilities and the problems of collecting and organizing the relevant data, which many railroad companies could not afford.

The situation has changed remarkably during the last years. Increasing computer speed and progress in mathematical methods enabled the development and solution of problem instances of more realistic models (for line planning problems, Bussieck [10] discussed these developments). As we have already mentioned, competition, privatization and deregulation require the efficient use of resources for the companies. This has affected air transportation companies to an especially large extent.

In Germany, the winter train schedule 1998/1999 was the first one to be developed with the help of computers completely (cf. [20]). However, this does not mean that the schedule was generated automatically, but with the help of decision support systems and graphical user interfaces.

A next step will be the simultaneous planning of several hierarchical levels, in the hope of achieving better overall solutions. In the Netherlands, the decision support system DONS (Designer of Network Schedules) assists the planners in routing *and* scheduling (cf. [38]). The CADANS module of DONS generates schedules, considering the railway infrastructure only from a global point of view. A second module, STATIONS, is responsible for checking whether a schedule is feasible with respect to the routing of trains through the railway stations, i.e. with the track layout. A comprehensive survey of discrete optimization techniques in public rail transport can be found in [9].

Besides optimization models, simulation tools for traffic planning are widely used to compare different scenarios for complex problems within short computing times.

## 1.1 Hierarchical Railroad Planning Levels

We will shortly focus on the different hierarchical planning steps from figure 1.2. Since these steps influence each other, it is of interest to discuss them and their connection to train scheduling to a certain extent. Our presentation follows [10] at this point.

### Passenger Demand

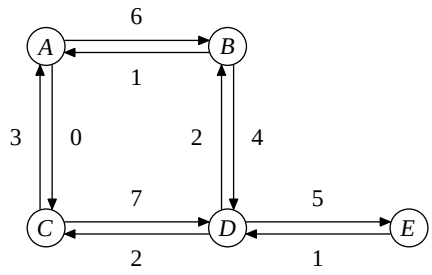
In order to establish a customer-oriented transportation service, the *passenger demand* or *traffic volume* must be given or estimated. The conventional form of passenger demand data is a so-called *origin destination matrix (OD-matrix)*. An entry $(i, j)$ of this matrix gives the number of people wishing to travel from location *i* to location $j$.

Sophisticated models and methods have been developed in order to determine OD-matrices. A number of cost-intensive *interviews of customers* may form a basis for statistical methods estimating the overall demand. Another approach are *traffic censuses* on the *network links* (like railroad tracks or streets). Statistical [40] and mathematical programming [58] methods that generate OD-matrices from such link traffic censuses are available. A disadvantage of this approach is that from the traffic volume on the links, OD-matrices with differently structured entries can be obtained. An example for this situation is given in figure 1.3. Another problem is that the routes taken by the travelers remain unknown. Nevertheless, OD-matrices are widely used in traffic planning models.

There is another general problem concerning the prediction of the passengers' behavior or wishes: The demand estimation based on methods like interviews or traffic census only reflects the current transportation service situation. If the line plan or train schedule is changed, passengers may change their behavior in an unpredictable way.

The main link between passenger demand and train scheduling is the problem of establishing train *connections* with adequate waiting times for travelers without a direct train from the origin to the destination of their trip. These travelers would like to have enough time to change the train (even in case of small delay), but of course do not wish to wait for a long time.

When establishing connections from OD-matrix data, one faces two main problems:




| | A | B | C | D | E |
|---|---|---|---|---|---|
| A | | 6 | | | |
| B | | | | | 4 |
| C | 1 | | | 7 | |
| D | 2 | 1 | | | 1 |
| E | 1 | | | | |

| | A | B | C | D | E |
|---|---|---|---|---|---|
| A | | | | 4 | |
| B | 1 | | | | |
| C | | 4 | | | 5 |
| D | 1 | | 2 | | |
| E | | | 1 | | |

Figure 1.3: Different OD-matrices for the same link traffic volume

- *Choice of routes*: As we have already mentioned, the routes of the travelers are not determined by the matrix. We may rely on the assumption that travelers mostly choose a shortest-path-like route for their trip. However, a short route concerning length in km may be served by a slower train or require an uncomfortable train change.
- *Choice of location for train change*: If passengers need to change between lines running on the same railroad track for some time, they may do so at several stations.

In these situations, *personal preferences* of the passengers play an important role, and objective decision criteria cannot be given. For train schedule planning, one should try to establish at least one "good" connection in this case.

### Line Planning

A line is given by a route and a corresponding frequency. The route is given by a path in the railroad track network. The frequency determines how often this line is served in accordance to the schedule period. *Line planning* means to select lines from a set of feasible lines subject to certain constraints and pursuing certain objectives.

Some possible constraints are that there must be enough lines (or trains respectively) to carry all passengers, the capacity of tracks must not be exceeded or that the required trains must be available. Common (and, as always, conflicting) objectives are minimization of costs or maximizing the number of travelers with a direct connection. Bussieck discusses line planning problems extensively in [10]. We will use and extend some concepts found in [10] in order to establish a new model for cost optimal train scheduling.

The periodicity of the schedule has to be kept in mind when designing line plans. In general, several trains (or so-called *train compositions*) are required in order to serve a line, because normally a train has not traveled the complete route and back in one schedule period.

Railroad companies usually offer different train service to their customers. In Germany for example, InterCity and InterCityExpress trains connect principal centers of the country. These trains are fast and equipped comfortably with dining car, phone or board services. InterRegio trains are slower and connect principal centers as well as district towns. Additionally, there are regional trains (like the AggloRegio trains in the Netherlands). All these trains have to share the global network.

For the planning process, the network is often decomposed into *supply networks* corresponding to the different train services (InterCity network, InterRegio network etc.). If a line plan for a single supply network has to be developed, the global demand information such as given by an OD-matrix has to be adapted for the supply networks. For example, there are approaches to split an OD-matrix into different matrices for the supply networks (*system split* procedure, cf. [50]).

The line plan serves as direct input for the train scheduling problem, where arrival and departure times for the lines have to be found. Furthermore, the line plan determines which travelers have to change a train during their trip and thus need acceptable connection times.

### Train Schedule Planning

The train schedule constitutes the backbone of public rail transport planning. The generation of train schedules is the core subject of this thesis. A detailed introduction to train scheduling problems is given in section 1.2.

A train schedule consists of the *arrival* and *departure times* of the lines at certain points of the network. Depending on the required *resolution,* these points are stations (low resolution) or even switches and important signal points (high resolution). For the railroad network of Germany, in the former case approximately 8000 such points are considered, in the latter case about 27000 points.

In general, schedules for public transport are periodical, i.e. the schedule is repeated after a *basic time period* or, for short, *period.*

The schedule fixes arrival and departure times for lines and thus for all trains of the line. An individual train corresponds to a *trip* of the line. The assignment of engines and coaches to these trips (or trains respectively) is done in a subsequent step.

### Planning of Rolling Stock and Crews

The trips established by the train schedule must be performed by some vehicles (motor unit, coaches) and crews (like engine drivers, conductors etc.). Optimization methods for vehicle scheduling in public transportation are described in [26, 39]. Since the dispatch of rolling stock and personnel has the main influence on the overall transport service costs, optimization methods are essential at this step, and other parts of the planning may have to be revised.

Crew management not only consists of dispatching train crews, but also local staff like cleaning staff or ticket office staff. Often, there are complex constraint systems for such duties, e.g. due to union contracts for break regulations or working times. Railway crew management experiences are reported in [11].

## 1.2 Train Schedule Planning: An Overview

We will start with a short historical introduction on train schedules (details can be found in [45]). In 1871, the first train schedule conference in Germany faced the difficult task of coordinating the schedules of the 80 railroad companies existing in Germany at that time (cf. [22]). The first schedules

introduced for long distance trains in the world were *non-periodic.* The reason might be that long distance trains were scheduled rarely (usually only one train per day), therefore a period would have been senseless. In highly congested urban areas, *periodic* or *fixed interval* schedules were used almost from the beginning (e.g. underground trains in London 1863, Budapest 1896, Paris 1900, Berlin 1902).

The main advantage of periodic schedules from customers' point of view is that they are easy to keep in mind. An example from [45] clearly shows this fact, see figure 1.4.

Schedule 1991/92

| hour | |
|---|---|
| 5 | **35** 52 |
| 6 | *33* **43** |
| 7 | **21** *33* **43** 58 |
| 8 | **38** 52 |
| 9 | *30* |
| 10 | 02 **43** |
| ⋮ | |

Schedule 1995/96

| hour | | | |
|---|---|---|---|
| 5 | **25** | 36 | **46** |
| 6 | *06* | 36 | **46** |
| 7 | *06* **25** | 36 | **46** |
| 8 | *06* | 36 | **46** |
| 9 | *06* **25** | 36 | **46** |
| 10 | *06* | 36 | **46** |
| ⋮ | | | |

departure for direction **Kaiserslautern**, Neustadt, *Saarbrücken*

Figure 1.4: Non-periodic and fixed interval schedule at Pirmasens

By introducing periodic schedules for long distance traffic in 1939, the railroad company of the Netherlands, *Nederlandse Spoorwegen*, marked a new epoch. Other European countries followed much later: Denmark in 1974, Switzerland in 1984, Belgium and Austria in 1991. In Germany, a fixed interval schedule for InterCity trains with a period of one hour was introduced in 1979. Beginning in 1985/86, InterRegio trains step by step were given a period of two hours. From 1992/93, also regional trains were scheduled in a fixed interval.

A further development is the *(perfect) integrated fixed interval schedule* (see [27]). This is a periodic schedule with specific *junction* points, where all trains serving that point arrive and depart nearly at the same time. Thus, at the junctions, transfer is possible between any pair of lines. If there is only one junction, such a schedule obviously always exists.

Traditionally, schedules are visualized by *time space diagrams* like in figure 1.5. In such diagrams, for a particular route of the network, all trains serving the route are represented. One can detect critical points or conflicts simply by looking at such a diagram: The trains speeds are represented by the respective gradients, and crossings indicate that trains overtake or encounter each other.

Besides the already mentioned train change times there are many other constraints for a schedule: The most important ones are *safety constraints.* Trains on the same track have to keep a certain *headway* distance. On network links with only a single track, trains must not start from different directions at the same time. Frequently used *objectives* for train schedule planning are the minimization of travel time, which mainly corresponds to a minimization of waiting time for train changes, minimization of certain costs or maximization of certain profits. A comprehensive introduction to constraints, objectives and models for train schedules is found in chapter 2.

In the last few years, computer software has been developed that is capable of effectively supporting the construction of schedules. Software products like ROMAN (ROute MANagement, this is used in

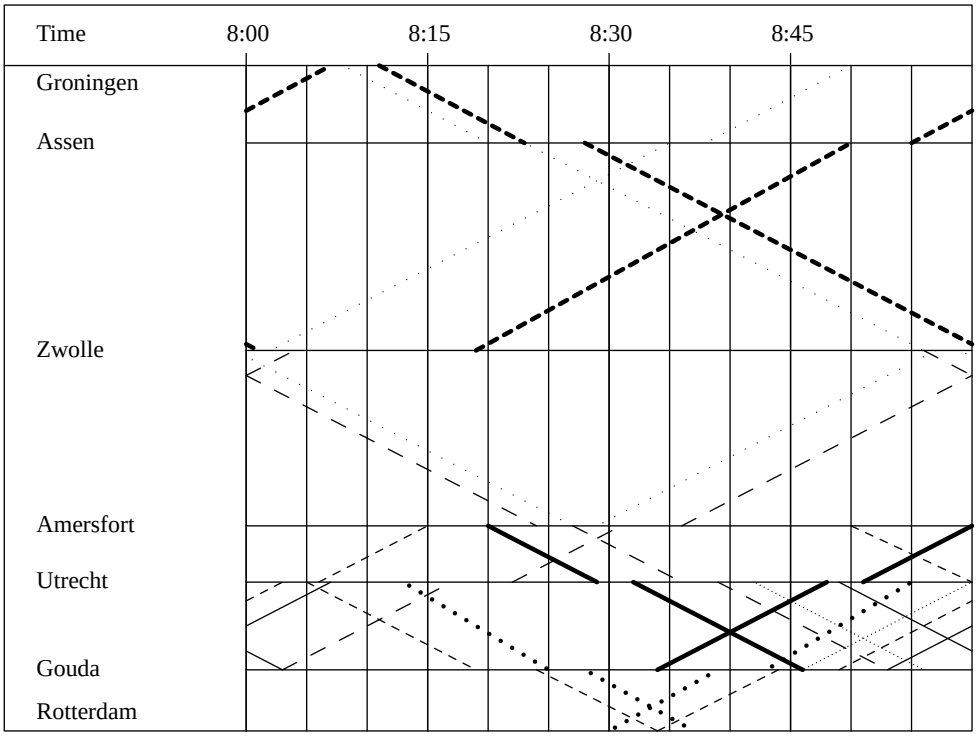



Figure 1.5: Time space diagram for a schedule on the route Groningen-Rotterdam

Germany and Austria) store information on track topology, engine and coach properties or available crews in databases. Thus, the running time of trains can be calculated in advance. Graphical user interfaces enable schedule planners to construct or edit schedules interactively based on time space diagrams like in figure 1.5. Conflicts (like missing headway) are automatically indicated on the screen. After the generation of a schedule, simulations can be performed.

However, with a few exceptions like the DONS system (which is mainly used for strategic planning), an automatic generation or even optimization of schedules is practically impossible at the moment. Most of the known algorithms are simply too slow for networks of practical size. Even worse, mathematical models for some aspects (like environmental effects, which will be a key aspect in the next few years, cf. [21]) still have to be developed.

# Chapter 2

# Models for Train Scheduling

In this chapter, mathematical models for the problems of generating and optimizing train schedules will be presented. The *periodic event scheduling problem (PESP)*, which is the problem of finding a feasible schedule subject to a particular class of constraints, forms a central part of the chapter. Several optimization criteria for train schedules are introduced, and a new model for cost optimal scheduling is presented. The model can be formulated as a mixed integer program. At the end of the chapter, the computational complexity of the problem of cost optimal scheduling is analyzed.

## 2.1 Railroad Networks and Train Schedules

A *railroad network* is usually represented by an undirected graph $G = (V, E)$, where $V$ is the set of nodes and $E$ is the set of edges. Depending on the required resolution, the nodes may represent stations or even switches and important signal points. The edges represent railroad tracks.

A *line* is modeled as a vector of nodes $(v_1, \ldots, v_n)$ with $v_i \in V$ for every $i \in \{1, \ldots, n\}$, $v_i \neq v_j$ for $i \neq j$, and $(v_i, v_{i+1}) \in E$ for every $i \in \{1, \ldots, n-1\}$. We always assume that all lines are served periodically with the same period $T \in \mathbb{N}$, i.e. we do not consider line frequencies. If there are lines with different periods, we may use the least common multiple of all periods as our single period, this will be discussed further below. The set of lines will be denoted by $\mathcal{R}$. Note that the elements of $\mathcal{R}$ are vectors of different dimensions.

Let $r = (v_1, \ldots, v_n) \in \mathcal{R}$ be a line. In our models, we assume that trains of this line run from $v_1$ to $v_n$ (via $v_2, \ldots$) and back to $v_1$ via $v_{n-1}, \ldots$ (this is not true in some real world cases: there exist lines using cycles instead of one path in both directions). We will use the notation $v \in r$ if there is a number $i \in \{1, \ldots, n\}$ such that $v = v_i$ and the notation $(v, v') \in r$ (and also $(v', v) \in r$) if there is a number $i \in \{1, \ldots, n-1\}$ such that $v = v_i$ and $v' = v_{i+1}$.

In general, the *events* that have to be scheduled are the arrivals or departures of lines at some locations represented by nodes $v \in V$. We consider *periodic events*, i.e. arrivals or departures of a line, and *individual events*, i.e. arrivals or departures of a particular train of a line. A formal definition is given now:

**Definition:** A *schedule* $\hat{\pi}$ for a set of *events* $\hat{\mathcal{E}}$ is a mapping $\hat{\pi} : \hat{\mathcal{E}} \to \mathbb{R}$. For an event $\hat{e} \in \hat{\mathcal{E}}$, $\hat{\pi}(\hat{e})$ is called the *event time* of $\hat{e}$. A *periodic event* $e$ is a countable set of (so-called *individual*) events $\{e^{(i)} \mid i \in \mathbb{Z}\}$ such that the event time $\hat{\pi}(e^{(i)})$ is given by $\hat{\pi}(e^{(0)}) + T \cdot i$.

By defining the event time for an individual event of a periodic event, the event times of all individual events of the periodic event are defined. For a set $\mathcal{E}$ of periodic events, let $\mathcal{E}^0 := \{e^{(0)} \mid e \in \mathcal{E}\}$. By assigning times to each element of $\mathcal{E}^0$, all times of individual events of the elements of $\mathcal{E}$ are assigned a time.

**Definition:** A *schedule* $\pi$ for a set of periodic events $\mathcal{E}$ is a mapping $\pi : \mathcal{E} \to \mathbb{R}$ defined by a mapping $\hat{\pi}$ for the corresponding individual events: $\pi(e) = x :\Leftrightarrow \hat{\pi}(e^{(0)}) = x$ for each $e \in \mathcal{E}$.

In our models, we will use the following notation for our periodic events:

| | |
|---|---|
| $a^v_{r,\mu}$ | arrival of line $r$, direction $\mu$, at station $v$ |
| $d^v_{r,\mu}$ | departure of line $r$, direction $\mu$, at station $v$ |

For simplicity, we will always assume that our graph nodes represent stations. The direction index may be 0 or 1 and is interpreted as follows: If $r = (v_1, \ldots, v_n)$, direction 0 means "on the way from $v_1$ to $v_n$", while direction 1 means on the way back. The index will be omitted if there can be no misunderstanding. The individual events of these periodic events correspond to the arrivals and departures of individual trains serving a line, i.e. the *trips*.

Many schedule constraints of practical interest can be formulated as so called *periodic interval constraints* for the periodic events ( [37, 48, 57] etc.). They have the following form:

$$\pi(e') \in \pi(e) + [l,u]_T :\Leftrightarrow \bigvee_{z \in \mathbb{Z}} \hat{\pi}(e^{(0)}) + l \leq \hat{\pi}(e'^{(0)}) - z \cdot T \leq \hat{\pi}(e^{(0)}) + u \tag{2.1}$$

with $e, e' \in \mathcal{E}$, $l, u \in \mathbb{R}$. We will also use the notation "$(e, e', l, u)$ is a periodic interval constraint" in order to express (2.1). Unions of periodic intervals can be modeled by intersections of periodic intervals (e.g. $[10,20]_{60} \cup [30,40]_{60} = [10,40]_{60} \cap [30,80]_{60}$).

Some examples for schedule constraints that can be modeled as periodic interval constraints are given here (cf. [37, 48, 57]):

- *Travel times*: Suppose that $(v, v') \in r$ and that $l$ is the minimum and $u$ the maximum allowed time for trains of line $r$ for the way from $v$ to $v'$. This can be expressed by the following constraint:

  $$\pi(a^{v'}_{r,0}) \in \pi(d^v_{r,0}) + [l,u]_T \tag{2.2}$$

  Note that, depending on the choice of $z$ from (2.1), individual events with different indices for arrival and departure may belong to the same train. A similar constraint for the other direction can be given easily. If the travel times are constant, we can set $l = u$.

- *Waiting times*: If the waiting time for line $r$ at station $v$ has to be in the interval $[l,u]$, the following constraint has to be satisfied:

  $$\pi(d^v_{r,0}) \in \pi(a^v_{r,0}) + [l,u]_T \tag{2.3}$$

- *Turnaround times*: If $r = (v_1, \ldots, v_n)$, we need a constraint of this form:

$$\pi(a^{v_n}_{r,1}) \in \pi(d^{v_n}_{r,0}) + [l,u]_T \tag{2.4}$$

  A turnaround time constraint for the other direction is not necessary because it is given implicitly by using periodic interval constraints.

- *Time for train changes*: As we have already discussed in section 1.1, those passengers with a change from one train to another one would like to have a certain connection time. This is provided by a constraint of this type:

$$\pi(d^{v}_{r',\mu'}) \in \pi(a^{v}_{r,\mu}) + [l,u]_T \tag{2.5}$$

  We have already seen that it is very difficult to determine such stations $v$ and lines $r, r'$. In section 5.4, we give a heuristic algorithm for determining lines and stations for train changes.

- *Headway times*: If $(v,v') \in r_1$ and $(v,v') \in r_2$ for $r_1, r_2 \in \mathcal{R}$ and there is only one railroad track leading from $v$ to $v'$, the trains of the lines $r_1$ and $r_2$ have to run on this same track. In order to avoid crashes, they should keep a certain headway distance (which is equivalent to a certain headway *time*). If the train speeds are constant (which is normally assumed for strategic and tactical planning models), the headway times only need to be guaranteed at the stations, leading to one periodic interval constraint for departure times and one constraint for arrival times:

$$\begin{aligned} \pi(d^{v}_{r_2,\mu}) &\in \pi(d^{v}_{r_1,\mu}) + [l,u]_T \\ \pi(a^{v'}_{r_2,\mu}) &\in \pi(a^{v'}_{r_1,\mu}) + [l,u]_T \end{aligned} \tag{2.6}$$

  An upper bound for the headway time is also necessary, because there has to be a headway time for preceding and for following trains.

  There are cases in which the headway constraints do not have the desired effect. This will be discussed in detail in section 2.5.

If there are lines $r_1, \ldots, r_m$ with periods $T_1, \ldots, T_m$, one can choose the least common multiple $T$ of $T_1, \ldots, T_m$ and replace each line $r_i$ by a set of *virtual* lines $r_{i,1}, \ldots, r_{i,T/T_i}$ whose departure and arrival times are connected by periodic interval constraints like

$$d^{v}_{r_{i,j+1},\mu} \in a^{v}_{r_{i,j},\mu} + [T_i, T_i]_T \tag{2.7}$$

This procedure presents another problem for the treatment of train changes. It is not known which of the virtual lines are used for the change. A constraint of the form

$$\bigvee_{i \in 1,\ldots,T/T_1} \; \bigvee_{j \in 1,\ldots,T/T_2} \pi(d^{v}_{r_{2,j},\mu_2}) \in \pi(a^{v}_{r_{1,i},\mu_1}) + [l,u]_T \tag{2.8}$$

needs to be satisfied, but this is not an interval constraint. Only in some special cases the constraint (2.8) can be transformed into a set of interval constraints.

**Proposition 2.1** *If travelers need to change from line $r_1$ with period $T_1$ to line $r_2$ with period $T_2$ at station $v$ with time interval $[l,u]$ (with $u-l<T_2$) and $T_1 = c\cdot T_2$ with $c\in\mathbb{N}$, then the following condition is equivalent to (2.8)*

$$\bigwedge_{q\in\{1,\dots,c\}} \pi(d^v_{r_{2,1},\mu_2}) \in \pi(a^v_{r_{1,1},\mu_1}) + [l+q\cdot T_2, u+T_1+(q-1)\cdot T_2]_{T_1}. \tag{2.9}$$

**Proof:** In order to simplify the notation, the indices for directions and stations are omitted. Suppose that (2.8) is true, i.e. there are $i\in\{1,\dots,T/T_1\}$, $j\in\{1,\dots,T/T_2\}$, $z_1,z_2,z_3\in\mathbb{Z}$ and $t\in[l,u]$ such that the following conditions hold:

$$\begin{aligned}\hat\pi(a^{(0)}_{r_{1,1}}) + (i-1)\cdot T_1 &= \hat\pi(a^{(0)}_{r_{1,i}}) - z_1\cdot T\\ \hat\pi(d^{(0)}_{r_{2,1}}) + (j-1)\cdot T_2 &= \hat\pi(d^{(0)}_{r_{2,j}}) - z_2\cdot T\\ \hat\pi(a^{(0)}_{r_{1,i}}) + t &= \hat\pi(d^{(0)}_{r_{2,j}}) - z_3\cdot T\end{aligned}$$

With $z := \frac{(-z_2+z_3+z_1)\cdot T}{T_1} + (i-1)$ (note that $z\in\mathbb{Z}$) this can be transformed to

$$\hat\pi(a^{(0)}_{r_{1,1}}) + t - (j-1)\cdot T_2 = \hat\pi(d^{(0)}_{r_{2,1}}) - z\cdot T_1.$$

Now determine $k^* := \min\{k\in\mathbb{Z} \mid k\cdot c-(j-1)\geq q\}$. It follows that $k^*\cdot c-(j-1)\geq q$, but $(k^*-1)\cdot c-(j-1)\leq q-1$. Because of

$$\begin{aligned}&\hat\pi(a^{(0)}_{r_{1,1}}) + l + q\cdot T_2 \leq \hat\pi(a^{(0)}_{r_{1,1}}) + t + (k^*\cdot c-(j-1))\cdot T_2 = \hat\pi(d^{(0)}_{r_{2,1}}) - (z-k^*)\cdot T_1\\ &= \hat\pi(a^{(0)}_{r_{1,1}}) + t + T_1 + ((k^*-1)\cdot c-(j-1))\cdot T_2 \leq \hat\pi(a^{(0)}_{r_{1,1}}) + u + T_1 + (q-1)\cdot T_2,\end{aligned}$$

the constraint (2.9) can be satisfied for every $q\in\{1,\dots,c\}$.

Conversely, let (2.9) be true. In this case, we have the following conditions:

$$\begin{aligned}\hat\pi(a^{(0)}_{r_{1,1}}) + t_1 &= \hat\pi(d^{(0)}_{r_{2,1}}) - z_1\cdot T_1 \quad t_1\in[l+T_2, u+c\cdot T_2]\\ \hat\pi(a^{(0)}_{r_{1,1}}) + t_2 &= \hat\pi(d^{(0)}_{r_{2,1}}) - z_2\cdot T_1 \quad t_2\in[l+2\cdot T_2, u+(c+1)\cdot T_2]\\ &\vdots\end{aligned}$$

Let $q^* := \max\{q\in\{1,\dots,c\} \mid t_1\geq l+q\cdot T_2\}$. Obviously $t_1\geq l+q^*\cdot T_2$ holds. We will now show that $t_1\leq u+q^*\cdot T_2$ is also true:

If $q^*=c$, the claim is correct from the discussion above. Let $q^*<c$. Since the intervals for the $t_i$ have a length $<T_1$, the choice of the $t_i$ and $z_i$ is uniquely determined. It can easily be seen that $z_q=z_1$ for $q\leq q^*$ and that $z_q=z_1-1$ for $q>q^*$. Now consider condition (2.9) for $q^*+1$. We have

$$\hat\pi(a^{(0)}_{r_{1,1}}) + t_{q^*+1} = \hat\pi(d^{(0)}_{r_{2,1}}) - z_{q^*+1}\cdot T_1,\ t_{q^*+1}\in[l+(q^*+1)\cdot T_2, u+(c+q^*)\cdot T_2].$$

Since $t_{q^*+1} = t_1 + z_1\cdot T_1 - z_{q^*+1}\cdot T_1 = t_1+T_1$, it follows that $t_1\in[l+(q^*+1)\cdot T_2 - T_1, u+q^*\cdot T_2]$.

Now we have shown that

$$\hat\pi(a^{(0)}_{r_{1,1}}) + t + q^*\cdot T_2 = \hat\pi(d^{(0)}_{r_{2,1}}) - z_1\cdot T_1 \text{ for a } t\in[l,u].$$

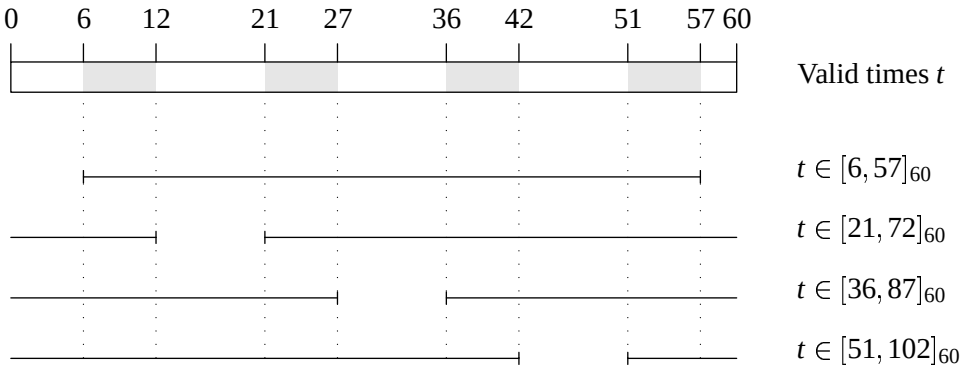



Figure 2.1: Valid changing times for $T = T_1 = 60$, $T_2 = 15$, $l = 6$, $u = 12$

From this, one can directly find the correct values to satisfy (2.8). □

An example illustration is given in figure 2.1.

There are many other aspects of railway scheduling which cannot be expressed as periodic interval constraints (for example constraints referring to individual trains). Some of them will be discussed later in this chapter.

## 2.2 The Periodic Event Scheduling Problem (PESP)

The problem of finding a schedule for periodic events subject to periodic interval constraints has been examined by several authors. In [59], Serafini and Ukovich defined the *Periodic Event Scheduling Problem (PESP)* similar to figure 2.2.

| Periodic Event Scheduling Problem (PESP): | | |
|---|---|---|
| Given: | $T$ | time period |
| | $\mathcal{E}$ | set of periodic events |
| | $\mathcal{C}$ | set of periodic interval constraints for $\mathcal{E}$ |
| Find: | $\pi : \mathcal{E} \to \mathbb{R}$ | schedule satisfying all constraints from $\mathcal{C}$ |
| | or state infeasibility | |

Figure 2.2: Periodic Event Scheduling Problem

Serafini and Ukovich proved that the PESP is NP-complete [59]. Odijk [49] proved that the problem is NP-complete even for fixed $T > 2$. More details on this can be found in section 2.9.

The PESP has been extensively examined by several authors (for example [42, 49, 57, 59]). Many of their algorithmic approaches to solve PESP instances will be discussed in chapter 3. Apart from train scheduling, the PESP has been applied to traffic light scheduling [60] and airline scheduling [30].

Moreover, the PESP is a basis for many schedule optimizing models. Some of them will be presented in section 2.6.

## 2.3 Event Graph Model

Often, the PESP is interpreted as a problem on the corresponding *PESP event graph*. For a PESP instance, the directed event graph $\mathcal{G} = (V_\mathcal{G}, A_\mathcal{G})$ is defined as follows:

- For each $e \in \mathcal{E}$, there is a node $v_e \in V_\mathcal{G}$.
- For each $(e, e', l, u) \in \mathcal{C}$, there is an arc from $v_e$ to $v_{e'}$ with a corresponding periodic interval $[l, u]_T$.

An example for a (part of a) network and the event graph to the corresponding PESP instance is given in figure 2.3. In the example case, only one direction of the lines is considered, so there are no direction indices.

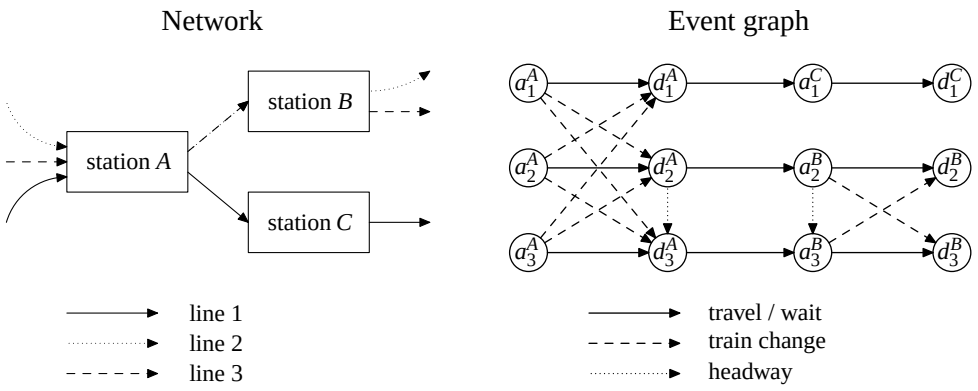



Figure 2.3: Network and event graph to the corresponding PESP instance

In the terminology of [59], a mapping $\varphi : V_\mathcal{G} \to \mathbb{R}$ is called *potential*. Every schedule $\pi$ for $\mathcal{E}$ represents a potential (and vice versa). For a potential $\varphi$, the corresponding mapping $\delta : A_\mathcal{G} \to \mathbb{R}$ defined by $\delta((v,v')) = \varphi(v') - \varphi(v)$ for each arc from $v$ to $v'$ is called *tension*.

A potential $\varphi$ (and the corresponding tension) is called *feasible*, if for each arc from $v_e$ to $v_{e'}$ in $A_\mathcal{G}$ representing the interval constraint $c = (e, e', l, u) \in \mathcal{C}$, there exists a $z_c \in \mathbb{Z}$ such that $\delta((v_e, v_{e'})) = t - z_c \cdot T$ for a $t \in [l, u]$, i.e. the respective schedule satisfies all periodic interval constraints. $z_c$ is called the *modulo parameter* for that particular arc.

For $a, b \in \mathbb{R}$, we define the following notation:

$$a \equiv b \bmod T :\Leftrightarrow \bigvee_{z \in \mathbb{Z}} b - a = z \cdot T$$

## 2.4 Linear Model with Integer Variables

The PESP can also be interpreted as the problem of finding a solution to a set of linear inequalities where some variables have to take an integer value. From figure 2.2 and (2.1), one can see that a

solution for $(\boldsymbol{\pi}, \boldsymbol{z})$ (where $\boldsymbol{\pi}$ is the vector of $\pi(e)$, $e \in \mathcal{E}$, and $\boldsymbol{z}$ is the vector of $z_c$, $c \in \mathcal{C}$) for the problem

$$\left\{ \begin{array}{rcrcll} l & \leq & \pi(e') - \pi(e) - z_c \cdot T & \leq & u & \text{for each } c = (e, e', l, u) \in \mathcal{C} \\ & & \pi(e) & \in & \mathbb{R} & \text{for each } e \in \mathcal{E} \\ & & z_c & \in & \mathbb{Z} & \text{for each } c \in \mathcal{C} \end{array} \right\} \tag{2.10}$$

is a solution for the PESP instance given by $T, \mathcal{E}, \mathcal{C}$.

Using the event graph formulation of the PESP with $\mathfrak{N}_{\mathcal{G}}$ as the node arc incidence matrix (see chapter 3 for details), $\boldsymbol{l}$ as the vector of lower and $\boldsymbol{u}$ as the vector of upper interval bounds, the linear system can be written as

$$\left\{ \begin{array}{rcrcl} \boldsymbol{l} & \leq & \mathfrak{N}_{\mathcal{G}}^T \boldsymbol{\pi} - T\boldsymbol{z} & \leq & \boldsymbol{u} \\ & & \boldsymbol{\pi} & \in & \mathbb{R}^{|V_{\mathcal{G}}|} \\ & & \boldsymbol{z} & \in & \mathbb{Z}^{|A_{\mathcal{G}}|} \end{array} \right\}. \tag{2.11}$$

There are algorithms based on this formulation (an example is Odijk's algorithm [47], which will be discussed in chapter 3).

Several constraints which cannot be expressed as periodic interval constraints can be given as linear constraints and thus can be added to the formulation (2.10) or (2.11). Some are given in section 2.5.

## 2.5 Extensions of the PESP

In this section we will discuss some extensions to the periodic interval constraint model, which will enable us to consider other practical schedule constraints.

### Single Track Connections

On single track connections (i.e. where the same single track is used for trains of both directions), trains must not start from different directions at the same time (for obvious reasons). To be exact, if there is a line $l_1$ running from $v$ to $v'$ and a line $l_2$ running from $v'$ to $v$ on the same track, the following constraints must be obeyed (direction indices are omitted):

$$\left\{ \begin{array}{rcll} \hat{\pi}(d_{l_2}^{v'(0)}) & \geq & \hat{\pi}(a_{l_1}^{v'(0)}) - z \cdot T & z \in \mathbb{Z} \\ \hat{\pi}(a_{l_2}^{v(0)}) & \leq & \hat{\pi}(d_{l_1}^{v(0)}) - z \cdot T + T & \text{for the same } z \end{array} \right\} \tag{2.12}$$

In other words, a train of line $l_2$ can only start after the arrival of a train of line $l_1$ in $v'$ and must arrive in $v$ before the next train of line $l_1$ departs there.

These constraints are not periodic interval constraints. Only if the travel times are constant, they can be transformed into interval constraints:

$$\left. \begin{array}{l} \hat{\pi}(a_{l_1}^{v'(0)}) = \hat{\pi}(d_{l_1}^{v(0)}) + t_1 \\ \hat{\pi}(a_{l_2}^{v(0)}) = \hat{\pi}(d_{l_2}^{v'(0)}) + t_2 \end{array} \right\} \Rightarrow \hat{\pi}(d_{l_1}^{v(0)}) + t_1 \leq \hat{\pi}(d_{l_2}^{v'(0)}) - z \cdot T \leq \hat{\pi}(d_{l_1}^{v(0)}) + T - t_2$$

The integer linear formulations (2.10) and (2.11) enable us to add single track connection constraints like (2.12) to the model by requesting $z = z'$ for the respective pairs of inequalities.

### Representative Trains

Since we have used periodic interval constraints for the travel, waiting and turnaround time of each line $l$, the corresponding individual events $d_{l,\mu}^{\nu(0)}$ or $a_{l,\mu}^{\nu(0)}$ do not necessarily belong to the same train. The same holds for different values of $\nu$.

Obviously, if there is a solution of a train scheduling PESP, then there is also a solution where all the individual events with index $(0)$ correspond to the same train (one may simply add suitable multiples of $T$ to the event times). Therefore, we will now always assume that in a PESP solution, individual events with index $(0)$ correspond to the same trains for each line. These trains will be called *representative trains.*

Using representative trains can be modeled by requesting $z = 0$ for the periodic interval constraints for traveling, waiting and turnaround time.

### Another Constraint Type for the Headway Problem

As we have mentioned, periodic interval constraints are not sufficient to provide correct headway times. An example is given now:

Let two lines $l_1$ and $l_2$ run on the same track vom $\nu$ to $\nu'$ (line direction indices will be omitted). Let $T = 60$ and the headway time $h = 2$ for trains of line $l_2$ following those of line $l_1$ and vice versa. Furthermore, let the travel times be given by $\pi(a_{l_1}^{\nu'}) \in \pi(d_{l_1}^{\nu}) + [20,22]_T$ and $\pi(a_{l_2}^{\nu'}) \in \pi(d_{l_2}^{\nu}) + [16,18]_T$. We assume that the trains have constant speed (which we do not know). Following section 2.1, we should introduce these constraints:

$$\pi(d_{l_2}^{\nu}) \in \pi(d_{l_1}^{\nu}) + [2,58]_T \qquad \text{and} \qquad \pi(a_{l_2}^{\nu'}) \in \pi(a_{l_1}^{\nu'}) + [2,58]_T$$

This does not lead to the desired result: A feasible schedule is given by

$$\hat{\pi}(d_{l_1}^{\nu(0)}) = 0, \quad \hat{\pi}(d_{l_2}^{\nu(0)}) = 2, \quad \hat{\pi}(a_{l_1}^{\nu'(0)}) = 20, \quad \hat{\pi}(a_{l_2}^{\nu'(0)}) = 18,$$

which means that a train of line $l_2$ has overtaken a train of line $l_1$ (remember that representative trains are considered), which may be impossible on the track from $\nu$ to $\nu'$. Figure 2.4 shows a corresponding time space diagram.

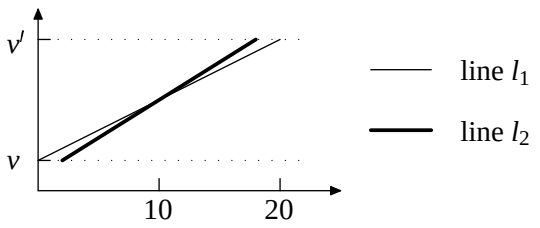



Figure 2.4: Time space diagram: a train passes the other one

With the help of the following proposition, we will derive a new type of constraint that will be helpful for handling the headway time problem.

**Proposition 2.2** *Let $l_1$ and $l_2$ be lines running on a railroad track from $v$ to $v'$. Assume that train speeds are constant and the following conditions hold (direction indices are omitted):*

$$\pi(d^v_{l_2}) \in \pi(d^v_{l_1}) + [l,u]_T \quad \text{with } 0 < l \leq u < T$$
$$\pi(a^{v'}_{l_2}) \in \pi(a^{v'}_{l_1}) + [l',u']_T \quad \text{with } 0 < l' \leq u' < T$$

*Then trains from different lines overtake each other if and only if for these constraints, condition (2.1) is satisfied for different values of z (with representative trains).*

**Proof:** We assume that it is possible to find a mapping from the railroad track from $v$ to $v'$ to the interval $[0,1]$ preserving continuity (this is always done when drawing time space diagrams, for example).

Let $\sigma_1(x)$ be the time at which the train related to to $d^{v(0)}_{l_1}$ passes the point corresponding to $x \in [0,1]$. Let $\sigma_2(x)$ be defined analogously for line $l_2$. Set $\sigma(x) = \sigma_2(x) - \sigma_1(x)$. Because of the constant train speeds, $\sigma$ is a monotone function on the interval $[0,1]$. It is also continous.

Trains from different lines overtake each other if and only if there is an $x \in (0,1)$ and a $k \in \mathbb{Z}$ for which $\sigma(x) = k \cdot T$ is true (this is obvious, because in this case, trains of different lines are at the same position on the track at the same time).

Now suppose that (2.1) is satisfied for both constraints with the same value of $z$, i.e.

$$\hat{\pi}(d^{v(0)}_{l_2}) = \hat{\pi}(d^{v(0)}_{l_1}) + zT + t \text{ with } t \in [l,u] \quad \text{and} \quad \hat{\pi}(a^{v'(0)}_{l_2}) = \hat{\pi}(a^{v'(0)}_{l_1}) + zT + t' \text{ with } t' \in [l',u']$$

for some $z \in \mathbb{Z}$. Then $\sigma(0) = zT + t$ and $\sigma(1) = zT + t'$. Since $\sigma$ is monotone, there cannot be an $x \in (0,1)$ for which $\sigma$ is a multiple of $T$.

Otherwise suppose that the condition is satisfied for different values $z$ and $z'$, i.e.

$$\hat{\pi}(d^{v(0)}_{l_2}) = \hat{\pi}(d^{v(0)}_{l_1}) + zT + t \text{ with } t \in [l,u] \quad \text{and} \quad \hat{\pi}(a^{v'(0)}_{l_2}) = \hat{\pi}(a^{v'(0)}_{l_1}) + z'T + t' \text{ with } t' \in [l',u']$$

Suppose $z < z'$ ($z > z'$ can be dealt with analogously). Now $\sigma(0) = zT - t$, $\sigma(1) = z'T - t'$, and because of the continuity of $\sigma$, there has to be an $x \in (0,1)$ with $\sigma(x) = z \cdot T$. □

We may now avoid train overtaking conflicts by demanding that for some pairs of interval constraints, the values of $z$ in condition (2.1) are equal (and representative trains are used). This leads to an extension of the PESP called *JPESP* (PESP with *joined constraints*), see figure 2.5.

### Disadvantages of Feasibility Models

The scheduling models discussed so far only consider feasibility. This leads to two main disadvantages for the practical use of algorithms based on those models:

- Practical instances may be infeasible. From a theoretical point of view, this does not present a problem, but in practice a schedule *has* to be generated. In order to make the instance feasible, some constraints have to be relaxed. But it is not at all clear which constraints should be relaxed or how they should be relaxed. A practical algorithm will have to decide that.
- If a schedule has been found, there is no information whether there are "better" schedules. In practice, there are many criteria for evaluating schedules (some of them will be mentioned in section 2.6). We will develop a new cost optimization model for train scheduling in section 2.8 (which will be based on a cost optimization model for line planning).

Periodic Event Scheduling Problem with joined constraints (JPESP):

| | | |
|---|---|---|
| Given: | $T$ | time period |
| | $\mathcal{E}$ | set of periodic events |
| | $\mathcal{C}$ | set of periodic interval constraints for $\mathcal{E}$ |
| | $\mathcal{J} \subseteq \mathcal{C} \times \mathcal{C}$ | set of joining conditions |
| Find: | $\pi : \mathcal{E} \to \mathbb{R}$ | schedule such that |
| | | • travel, waiting, turnaround constraints are satisfied with $z = 0$ |
| | | • all other constraints from $\mathcal{C}$ are satisfied with arbitrary $z \in \mathbb{Z}$ |
| | | • $(c_1, c_2) \in \mathcal{J} \Rightarrow z_{c_1} = z_{c_2}$ |
| | or state infeasibility | |

Figure 2.5: Periodic Event Scheduling Problem with joined constraints

## 2.6 Schedule Optimization Models

Railroad companies have many different (and conflicting) optimization criteria for schedules, including:

- *Minimization of total travel time for passengers*: An important aspect determining the attractiveness of a schedule is to keep the trip times for passengers short. Since train speeds often cannot be varied much, the largest optimization potential here comes from the waiting times for passengers who need to change from one train to another. In case of variable waiting times, these times should also be kept as small as possible.

  Let $\overline{\mathcal{C}} \subseteq \mathcal{C}$ be the set of train change time constraints from section 2.1. Suppose that for every $c \in \overline{\mathcal{C}}$, the number of passengers $\omega_c$ who need the respective connection is known (as we have pointed out in section 1.1, it is difficult to determine these numbers). Then the sum of waiting times for all passengers is given by

$$\sum_{\left(a^v_{l_1,\mu_1}, d^v_{l_2,\mu_2}, l, u\right) = c \in \overline{\mathcal{C}}} \omega_c \cdot \left(\pi(d^v_{l_2,\mu_2}) - \pi(a^v_{l_2,\mu_2}) - z_c \cdot T\right). \tag{2.13}$$

  This is a linear expression in the values of $\pi(e)$ and thus can be added to the PESP formulation in order to get a mixed integer linear program (MIP). In [37], Krista solved this MIP (with an additional cost term for train waiting time very similar to (2.13)) for several real world network instances with a commercial MIP solver. Nachtigall [41,42] developed a branch-and-cut method to solve the problem.

  An alternative approach for minimizing the train change time is given in [19]. There, Daduna and Voß used a quadratic semi assignment model and a tabu search heuristic. Kolonko et al. look for *pareto optimal* solutions concerning minimum trip time and investment cost for upgrading the network tracks [36]. In this case, a greedy heuristic and a genetic algorithm are used.

- *Maximization of robustness in case of train delays*: In practice, train delays occur very often. In this situation, other trains using the same track may have to wait, and so the overall system delay increases in a cascade-like process. Furthermore, if passengers arrive at their train changing station late, they may lose their connection. Alternatively, other trains have to wait, and again the total delay increases. To avoid this, one can try to maximize the minimum headway of trains arriving or departing at the same point in the network. As a consequence, all trains have a headway that is larger than actually required. In case of delays, the corresponding constraints may be disobeyed, as long as the actually required headway is guaranteed.

  This approach has been followed by Heusch et al. in [31], where a generalized graph coloring model and a corresponding backtracking algorithm is presented.

- *Maximization of profit / minimization of costs*: There are several ideas for estimating the profit / cost of a train schedule, resulting in different models for schedule optimization:

  Brännlund et al. developed a model for a profit maximizing schedule in [6]. In their model, the profit depends on the time that certain trains pass certain parts of the network. They formulate a binary variable linear program and give heuristic solutions by a Lagrangian relaxation method.

  In [12, 13], Carey considers a minimal cost scheduling model, where there are trip time costs, dwell time costs or costs for special arrival and departure times. The model results in a binary variable linear program, which is treated heuristically.

  Another model introduced by Higgins et al. in [32] uses a weighted sum of delay and train operating costs. On their binary program, a special branch-and-bound method with nonlinear subproblems is used. The model is only used on single line rail corridors.

  In section 2.8, we will introduce a new model for minimal cost train scheduling, which is based on a cost model for line planning.

- *Minimization of the period*: There may be situations in which the minimal possible period for a traffic system is of interest. This approach is somewhat different from the others, since for this problem, the period is variable. In [7,52], the problem is formulated and solved as an eigenvalue problem for the maxplus-algebra.

An overview on optimization models for train routing and scheduling can be found in [18].

Another idea for schedule optimization is the "minimization of the infeasibility" of a PESP instance. Let $\pi$ be a schedule (which may be infeasible) for a PESP instance. Then for each $c = (e, e', l, u) \in \mathcal{C}$, the *constraint violation* $\varepsilon_c$ is defined by

$$\varepsilon_c := \min_{z_c \in \mathbb{Z}} \max\{0, \pi(e') - \pi(e) - z_c \cdot T - u, l - (\pi(e') - \pi(e) - z_c \cdot T)\}. \tag{2.14}$$

Some possibilities for "infeasibility minimization" using $\varepsilon_c$ are:

- Find a schedule $\pi$ such that the number of constraints $c$ with $\varepsilon_c > 0$ is minimized.
- Find a schedule $\pi$ such that $\sum_{c \in \mathcal{C}} \varepsilon_c$ is minimized.
- Find a schedule $\pi$ such that $\max_{c \in \mathcal{C}} \varepsilon_c$ is minimized.

## 2.7 Cost Model for Line Planning

In this section, we will describe a model for cost-oriented line planning. Based on this model, we will develop a cost-oriented model for train scheduling in section 2.8. The line planning model was introduced by Claessens in [16] in cooperation with the Dutch railroad company Nederlandse Spoorwegen (NS) and Railned (a Dutch state organization responsible for capacity planning, management of the infrastructure and for railroad safety). It was further examined and tested on practical networks of the Netherlands in [17] and [10].

Given a network $G = (V,E)$, a set of possible lines $\mathcal{R}^P$ and a set of possible frequencies $\mathcal{F}_r$ for each $r \in \mathcal{R}^P$, the line optimization is to find a subset $\mathcal{R} \subseteq \mathcal{R}^P$ and frequencies $f_r \in \mathcal{F}_r$ for each $r \in \mathcal{R}$ such that certain constraints are satisfied and a certain objective is minimized.

In the model proposed in [16], not only lines and frequencies are determined, but also numbers of coaches for the trains (i.e. trains do not have a fixed length a priori). The following cost aspects are considered by the model:

- *Fixed cost per schedule period per motor unit and per coach*: This includes depreciation cost, capital cost, fixed maintenance cost or cost for overnight parking.
- *Cost per km per motor unit and per coach*: Examples are energy and maintenance cost.

In order to determine the cost of a schedule period of a particular line $r \in \mathcal{R}^P$, we need to know the number of train compositions required for operating the line and the distance the trains have to run.

Let $r \in \mathcal{R}^P$, let $f_r \in \mathcal{F}_r$ be a frequency for $r$ and let $t_r$ be the time required by a train to fulfill a complete circulation. Since this time may depend on the actual schedule and thus is not known exactly in advance, an estimation $\hat{t}_r$ is used. The number of trains required for the line is then given by

$$\hat{\gamma}_r := \left\lceil \frac{f_r \cdot \hat{t}_r}{T} \right\rceil . \tag{2.15}$$

Let $d_r$ be the length of a circulation of line $r$. During a schedule period, the sum of the distances run by all trains of line $r$ is $d_r \cdot f_r$ (which is independent of $\hat{\gamma}_r$, as one can easily verify!).

An example for the calculation of $\hat{\gamma}_r$ is given in figure 2.6.

The following types of constraints are considered in [16]:

- *Numbers of coaches*: For each line $r \in \mathcal{R}^P$, there is a lower and an upper bound for the number of coaches.
- *Line frequency for edges*: For each network edge, there is a lower and an upper bound for the sum of the frequencies of lines running on that edge.
- *Traveler capacity*: On each network edge, there is a lower bound for the sum of the traveler capacities of the trains running on that edge in one schedule period.

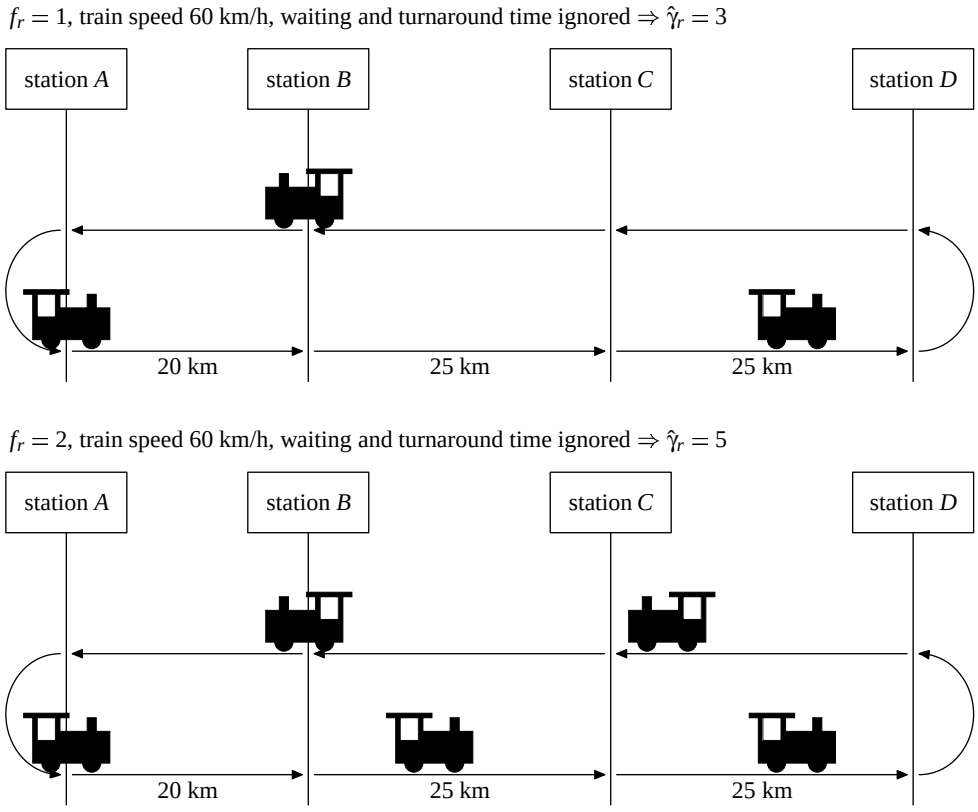



Figure 2.6: Circulation of trains for different frequencies

In [16], a nonlinear integer program is constructed to solve the problem. We will give a slightly modified version of the formulation here. The variables are:

$x_r \in \mathcal{F}_r$ frequency of line $r \in \mathcal{R}^P$

$w_r \in \mathbb{Z}$ number of coaches for the trains of line $r \in \mathcal{R}^P$

With the notation of table 2.1 for the input data, the model is given in figure 2.7.

The results obtained with this model and a heuristic solution procedure are reported to be quite unsatisfactory (cf. [16]). Better results were produced with two kinds of linearizations of the COSTNLP model. They will be discussed now.

Instead of using integer variables for the frequency, in [10] binary variables are introduced indicating that a certain frequency is used or not used. Furthermore, for each feasible frequency for a line, an integer variable for the number of coaches is used:

$x_{r,f} \in \{0,1\}$ line $r \in \mathcal{R}^P$ is used with frequency $f \in \mathcal{F}_r$

$w_{r,f} \in \mathbb{Z}$ number of coaches in trains of frequency $f \in \mathcal{F}_r$ for line $r \in \mathcal{R}^P$

This substitution leads to a linear integer programming model, figure 2.8. There, some constraints have to be added to ensure that only one frequency is used for a line and that no coaches are used if the corresponding frequency is not selected.

| | | | |
|---|---|---|---|
| $C^{\text{fix}}$ | fixed cost per motor unit | $C^{\text{fixC}}$ | fixed cost per coach |
| $C^{\text{km}}$ | km cost per motor unit | $C^{\text{kmC}}$ | km cost per coach |
| $d_r$ | circulation length of line $r$ | $\hat{t}_r$ | estimated circulation time of line $r$ |
| $\underline{W}$ | min. # coaches per train | $\overline{W}$ | max. # coaches per train |
| $\underline{lfr}_e$ | min. line frequency for edge $e$ | $\overline{lfr}_e$ | max. line frequency for edge $e$ |
| $N_e$ | # travelers on edge $e$ | $\mathfrak{C}$ | coach capacity |
| $T$ | time period | | |

Table 2.1: Parameters for cost-related line optimization

Nonlinear integer program for cost-related line optimization (COSTNLP):

$$\min \quad \sum_{r\in\mathcal{R}^P} \lceil x_r\cdot\hat{t}_r/T\rceil\cdot(C^{\text{fix}}+w_r\cdot C^{\text{fixC}})+x_r\cdot d_r\cdot(C^{\text{km}}+w_r\cdot C^{\text{kmC}})$$

$$\begin{aligned}
\underline{lfr}_e \;\le\; \sum_{r\in\mathcal{R}^P,r\ni e} x_r \;&\le\; \overline{lfr}_e && \text{for each } e\in E\\
\mathfrak{C}\cdot\sum_{r\in\mathcal{R}^P,r\ni e} x_r\cdot w_r \;&\ge\; N_e && \text{for each } e\in E\\
\underline{W} \;\le\; w_r \;&\le\; \overline{W} && \text{for each } r\in\mathcal{R}^P\\
x_r \;&\in\; \mathcal{F}_r && \text{for each } r\in\mathcal{R}^P\\
w_r \;&\in\; \mathbb{Z} && \text{for each } r\in\mathcal{R}^P
\end{aligned}$$

Figure 2.7: Nonlinear integer program for cost-related line optimization

Integer linear program for cost-related line optimization (COSTILP):

$$\min \quad \sum_{r\in\mathcal{R}^P}\sum_{f\in\mathcal{F}_r} \lceil f\cdot\hat{t}_r/T\rceil\cdot(x_{r,f}\cdot C^{\text{fix}}+w_{r,f}\cdot C^{\text{fixC}})+f\cdot d_r\cdot(x_{r,f}\cdot C^{\text{km}}+w_{r,f}\cdot C^{\text{kmC}})$$

$$\begin{aligned}
\underline{lfr}_e \;\le\; \sum_{r\in\mathcal{R}^P,r\ni e}\sum_{f\in\mathcal{F}_r} f\cdot x_{r,f} \;&\le\; \overline{lfr}_e && \text{for each } e\in E\\
\mathfrak{C}\cdot\sum_{r\in\mathcal{R}^P,r\ni e}\sum_{f\in\mathcal{F}_r} f\cdot w_{r,f} \;&\ge\; N_e && \text{for each } e\in E\\
\underline{W}\cdot x_{r,f} \;\le\; w_{r,f} \;&\le\; \overline{W}\cdot x_{r,f} && \text{for each } r\in\mathcal{R}^P \text{ and } f\in\mathcal{F}_r\\
\sum_{f\in\mathcal{F}_r} x_{r,f} \;&\le\; 1 && \text{for each } r\in\mathcal{R}^P\\
x_{r,f} \;&\in\; \{0,1\} && \text{for each } r\in\mathcal{R}^P \text{ and } f\in\mathcal{F}_r\\
w_{r,f} \;&\in\; \mathbb{Z} && \text{for each } r\in\mathcal{R}^P \text{ and } f\in\mathcal{F}_r
\end{aligned}$$

Figure 2.8: Integer linear program for cost-related line optimization

For this model, several preprocessing techniques and a cutting plane algorithm have been developed in [10]. For several practical optimization instances of Nederlandse Spoorwegen, high quality so-

lutions (i.e. solutions with small MIP gaps or even optimal solutions) could be found by using the COSTILP model (with preprocessing, cutting planes and the use of the commercial modeling system GAMS (cf. [23] and [24]) and the commercial MIP solver CPLEX [34]).

Another linearization approach for COSTNLP is found in [17]. In this case, not only binary variables are used as frequency indicators but also for the numbers of coaches:

$$w_{r,f,c} \in \{0,1\} \quad \text{line } r \in \mathcal{R}^P \text{ is used with frequency } f \in \mathcal{F}_r \text{ and } c \text{ coaches}$$

This method results in a linear program with (a lot of) binary variables, see figure 2.9.

Binary variable linear program for cost-related line optimization (COSTBLP):

$$\min \quad \sum_{r\in\mathcal{R}^P} \sum_{f\in\mathcal{F}_r} \sum_{c=\underline{W}}^{\overline{W}} \left( \lceil f \cdot \hat{t}_r/T \rceil \cdot (C^{\text{fix}} + c \cdot C^{\text{fixC}}) + f \cdot d_r \cdot (C^{\text{km}} + c \cdot C^{\text{kmC}}) \right) \cdot w_{f,r,c}$$

$$\begin{aligned}
\underline{lfr}_e \quad \leq \quad & \sum_{r\in\mathcal{R}^P, r\ni e} \sum_{f\in\mathcal{F}_r} \sum_{c=\underline{W}}^{\overline{W}} f \cdot w_{r,f,c} \quad \leq \quad \overline{lfr}_e \qquad \text{for each } e \in E \\
\mathfrak{C} \cdot \sum_{r\in\mathcal{R}^P, r\ni e} \sum_{f\in\mathcal{F}_r} \sum_{c=\underline{W}}^{\overline{W}} & f \cdot c \cdot w_{r,f,c} \quad \geq \quad N_e \qquad \text{for each } e \in E \\
\sum_{f\in\mathcal{F}_r} \sum_{c=\underline{W}}^{\overline{W}} & w_{r,f,c} \quad \leq \quad 1 \qquad \text{for each } r \in \mathcal{R}^P \\
& w_{r,f,c} \in \{0,1\} \text{ for each } r \in \mathcal{R}^P,\ f \in \mathcal{F}_r \text{ and } c \in \{\underline{W}, \ldots, \overline{W}\}
\end{aligned}$$

Figure 2.9: Binary variable linear program for cost-related line optimization

The solutions generated by this model were not as good as those for COSTILP. As reported in [10], on the one hand the binary variables provided a better LP relaxation, while on the other hand, the branch-and-bound process for MIP solving was slowed down by the enormous size of the problem.

In [10], COSTILP was extended to cover several supply networks (for example InterCity and Inter-Regio network) simultaneously. Apart from larger problem sizes, other practical problems may occur for such models:

- The model may select cheaper, but slower and therefore less attractive train types for many lines.
- Interactions between the train types are difficult to control (for example train speeds).

## 2.8 Cost Model for Train Scheduling

We will now assemble the ideas from feasibility models for schedules and for cost optimization for lines to get a new model for *cost optimal scheduling.* Suppose that a line plan has been found (i.e. $\mathcal{R} \subseteq \mathcal{R}^P$ has been selected). In our model, we assume that the railroad company still is faced the task of assigning *train types* to the lines. A train type is characterized by:

- cost, capacity of coaches, bounds on number of coaches (as in the line planning model)
- speed

The possibility of choosing from a set of train types may result from the fact that the supply networks for the lines are not fixed in advance (although this may cause difficulties as mentioned at the end of section 2.7) or that there actually are several motor units and coaches with different properties available for the same supply network. Let $\mathcal{T}$ denote the set of train types.

The choice of train types influences the schedule via the speed. Let $\mathcal{T}_r \subseteq \mathcal{T}$ be the set of feasible train types for line $r$. Then the travel time constraints for line $r$ are relaxed from

$$\pi(a^{v'}_{r,\mu}) \in \pi(d^{v}_{r,\mu}) + [l,u]_T \qquad \text{to} \qquad \bigvee_{\tau\in\mathcal{T}_r} \pi(a^{v'}_{r,\mu}) \in \pi(d^{v}_{r,\mu}) + [l_\tau,u_\tau]_T.$$

Only for some combinations of train types, there will be a feasible schedule. Our model will determine the minimum cost train type combination (including numbers of coaches) for which a feasible schedule exists.

Since the COSTILP model gave the best results for the line optimization problem, we will develop a similar integer linear programming model for the scheduling problem. Our variables are (for a short notation, we now use $a$ and $d$ for event times instead of events):

| | |
|---|---|
| $x_{r,\tau}$ | train type $\tau \in \mathcal{T}_r$ is used for line $r \in \mathcal{R}$ |
| $w_{r,\tau}$ | number of coaches of type $\tau$ for trains of line $r \in \mathcal{R}$ |
| $a^{v}_{r,\mu}$ | arrival time of individual train $(0)$, direction $\mu$ of line $r$ in $v$ |
| $d^{v}_{r,\mu}$ | departure time of individual train $(0)$, direction $\mu$ of line $r$ in $v$ |
| $\boldsymbol{z}$ | vector of modulo parameters for JPESP constraints |

The constants form the line optimization models are given an additional index for the train type if necessary. The travel time bounds now depend on the train types:

| | |
|---|---|
| $\underline{trav}^{vv'}_{\tau}$ | minimum travel time for trains of type $\tau$ from $v$ to $v'$ |
| $\overline{trav}^{vv'}_{\tau}$ | maximum travel time for trains of type $\tau$ from $v$ to $v'$ |
| $\underline{wait}, \overline{wait}$ | minimum and maximum waiting time |
| $\underline{turn}, \overline{turn}$ | minimum and maximum turnaround time |

Of course, the time bounds may depend on other criteria as well. The complete MIP model for the *minimum cost scheduling problem (MIP-MCSP)* is given in figure 2.10. Note that:

- in order to avoid a nonlinear model, we still use estimations (depending on the train type) for the circulation time,
- we assume that $\mathcal{R}$ is the set of lines after having introduced a common period (i.e. all lines have the same frequency),
- in the travel time constraint, $\mu$ is the direction in which node $v$ is directly followed by $v'$,

Mixed integer linear program for minimum cost scheduling (MIP-MCSP):

$$\min \sum_{r\in\mathcal{R}} \sum_{\tau\in\mathcal{T}_r} \lceil \hat{t}_{r,\tau}/T \rceil \cdot (x_{r,\tau}\cdot C_\tau^{\text{fix}} + w_{r,\tau}\cdot C_\tau^{\text{fixC}}) + d_r \cdot (x_{r,\tau}\cdot C_\tau^{\text{km}} + w_{r,\tau}\cdot C_\tau^{\text{kmC}})$$

$$\begin{aligned}
\sum_{r\in\mathcal{R}, r\ni e} \sum_{\tau\in\mathcal{T}_r} \mathfrak{C}_\tau \cdot w_{r,\tau} &\geq N_e && \text{for each } e\in E\\
\underline{W}_\tau \cdot x_{r,\tau} \leq w_{r,\tau} &\leq \overline{W}_\tau \cdot x_{r,\tau} && \text{for each } r\in\mathcal{R} \text{ and } \tau\in\mathcal{T}_r\\
\sum_{\tau\in\mathcal{T}_r} x_{r,\tau} &= 1 && \text{for each } r\in\mathcal{R}\\
\\
\sum_{\tau\in\mathcal{T}_r} \underline{trav}_\tau^{vv'} x_{r,\tau} \leq a_{r,\mu}^{v'} - d_{r,\mu}^{v} &\leq \sum_{\tau\in\mathcal{T}_r} \overline{trav}_\tau^{vv'} x_{r,\tau} && \text{for each } r\in\mathcal{R}, (v,v')\in r\\
\underline{wait} \leq d_{r,\mu}^{v} - a_{r,\mu}^{v} &\leq \overline{wait} && \text{for each } r\in\mathcal{R}, v\in r, \mu\\
\underline{turn} \leq a_{r,1}^{v_n} - d_{r,0}^{v_n} &\leq \overline{turn} && \text{for each } r=(v_1,\ldots,v_n)\in\mathcal{R}
\end{aligned}$$

other JPESP constraints

$$\begin{aligned}
x_{r,\tau} &\in \{0,1\} && \text{for each } r\in\mathcal{R} \text{ and } \tau\in\mathcal{T}_r\\
w_{r,\tau} &\in \mathbb{Z} && \text{for each } \tau\in\mathcal{R} \text{ and } \tau\in\mathcal{T}_r\\
a_{r,\mu}^{v} &\in \mathbb{R} && \text{for each } r\in\mathcal{R}, v\in r, \mu\\
d_{r,\mu}^{v} &\in \mathbb{R} && \text{for each } r\in\mathcal{R}, v\in r, \mu\\
\boldsymbol{z} && \text{integer vector for corresponding JPESP dimension}
\end{aligned}$$

Figure 2.10: Mixed integer linear program for minimum cost scheduling

- because of the periodicity, only one turnaround constraint per line is needed.

The model consists of two blocks of constraints which are only connected by the train type variables. The first three classes of constraints are very similar to the line optimization constraints. The remaining classes form a JPESP if the train types are fixed. We will use these blocks for a decomposition method in section 4.2.

If the travel time estimation $\hat{t}_{r,\tau}$ is replaced by $d_{r,1}^{v_1} - a_{r,0}^{v_1} + \underline{turn}$ for each $r=(v_1,\ldots,v_n)\in\mathcal{R}$, the model becomes exact, but nonlinear. In section 4.7, we will develop an algorithm for solving this nonlinear problem (which will be, however, much too slow for problem instance sizes of practical interests).

## 2.9 Computational Complexity

In this section, we will give an overview on computational complexity results on the PESP. Additionally, we will examine the complexity of the cost optimal scheduling problem from section 2.8. Some remarks on complexity can be found in appendix A, more details on this subject are given in [25].

### 2.9.1 Complexity Results on the PESP

For the following theorems, we assume that we are given integral values as interval bounds and period. Also the schedules are expected to be integer valued.

**Theorem 2.1** *The PESP is* NP*-complete.*

A proof is given in [59]. The Hamilton cycle problem (HCP), which is NP-complete, can be polynomially transformed to the PESP. The HCP is the problem to determine whether a non-directed graph contains a cycle covering all vertices exactly once. The problem of the proof is that the period $T$ depends on the size of the HCP instance.

For practical instances, there is always a fixed period (e.g. 60 for hourly trains). Therefore, it is interesting to examine the complexity of the PESP for fixed $T$.

**Theorem 2.2** *The PESP is in* P *for* $T = 2$.

In [45], Nachtigall proved this theorem. There are only two reasonable interval bounds in case of $T = 2$: $[0,0]$ and $[1,1]$. If there is a constraint $c = (e,e',0,0) \in \mathcal{C}$ it follows that $\pi(e') \equiv \pi(e) \bmod 2$, otherwise $\pi(e') \not\equiv \pi(e) \bmod 2$. The existence of a schedule for such an instance can obviously checked by a simple labeling procedure working with complexity $O(|\mathcal{C}|)$.

**Theorem 2.3** *The PESP is* NP*-complete for fixed* $T > 2$.

Odijk [47] shows that instances of the $K$-colorability problem for fixed $K$, which is known to be NP-complete [25], can be polynomially transformed to instances of the PESP with period $K$. The $K$-colorability problem is the problem of determining, whether for a given graph $G = (V,E)$ and a number $K \in \mathbb{N}$, $K < |V|$, there is a mapping $c : V \to \{1,\ldots,K\}$ such that $c(v) \neq c(v')$ whenever $(i,j) \in E$.

To a given $K$-colorability problem instance $G = (V,E)$, construct a PESP instance $\mathcal{G} = (V,E')$, where $E'$ is obtained from $E$ by choosing arbitrary directions for the elements $e \in E$. Set $T = K$ and take interval bounds $l = 1$, $u = T - 1$ for every arc. Obviously, there is a one-to-one correspondence between feasible potentials for $\mathcal{G}$ and feasible colorings of $G$.

### 2.9.2 Complexity Results on Cost Optimal Scheduling

The minimum cost scheduling problem discussed in section 2.8 contains several difficult aspects. As we have seen in section 2.9.1, finding a feasible schedule for a set of periodic interval constraints is already NP-complete. We will now focus on the selection of train types, which will also lead to NP-complete problems.

We will now consider the minimum cost scheduling problem without the scheduling constraint (i.e. we are only concerned with the first block of constraints in figure 2.10). This problem will be called *minimum cost type problem (MCTP)*. Furthermore, we focus on a decision version of the problem, namely determining whether there is a choice of train types and numbers of cars such that all constraints are satisfied and the objective function does not exceed a certain value. This problem will be called *Decision-MCTP* and is given in figure 2.11 independently of a model (such as MIP-MCSP).

Decision version of the minimum cost type problem (Decision-MCTP):

| | | |
|---|---|---|
| Given: | $G = (V, E)$ | network graph |
| | $\mathcal{R}$ | set of lines |
| | $\mathcal{T}$ | set of train types |
| | $\mathcal{T}_r \subseteq \mathcal{T}$ | set of feasible train types for each $r \in \mathcal{R}$ |
| | $C^{\text{fix}}, C^{\text{fixC}}, C^{\text{km}}, C^{\text{kmC}} : \mathcal{T} \to \mathbb{N}_0$ | cost functions |
| | $\underline{W}, \overline{W} : \mathcal{T} \to \mathbb{N}$ | bounds for numbers of coaches |
| | $d : \mathcal{R} \to \mathbb{N}$ | length of line circulation |
| | $\hat{\gamma} : \mathcal{R} \times \mathcal{T} \to \mathbb{N}$ | estimated number of train compositions |
| | $\mathfrak{C} : \mathcal{R} \to \mathbb{N}$ | coach capacity |
| | $N : E \to \mathbb{N}$ | number of travelers |
| | $K \in \mathbb{N}$ | cost limit |
| Find: | $x : \mathcal{R} \to \mathcal{T}$ and $w : \mathcal{R} \to \mathbb{N}$ such that | |
| | | 1. $x(r) \in \mathcal{T}_r$ for each $r \in \mathcal{R}$ |
| | | 2. $\underline{W}(x(r)) \leq w(r) \leq \overline{W}(x(r))$ for each $r \in \mathcal{R}$ |
| | | 3. $\sum_{r \in \mathcal{R}:\ r \ni e} \mathfrak{C}(x(r)) \cdot w(r) \geq N(e)$ for each $e \in E$ |
| | | 4. $\sum_{r \in \mathcal{R}} \hat{\gamma}(r, x(r)) \cdot \left(C^{\text{fix}}(x(r)) + w(r) \cdot C^{\text{fixC}}(x(r))\right)$ $+ d(r) \cdot \left(C^{\text{km}}(x(r)) + w(r) \cdot C^{\text{kmC}}(x(r))\right) \leq K$ |
| | or state infeasibility | |

Figure 2.11: Decision version of the minimum cost type problem

**Theorem 2.4** *The Decision-MCTP is* NP*-complete.*

**Proof:** The problem is in NP: A solution consists of the values of $x$ and $w$ and is therefore polynomially bounded in the input size. The properties 1–4 can of course be checked in polynomial time.

We will now show that instances of the knapsack problem of figure 2.12, which is known to be NP-complete [25], can be polynomially transformed to instances of the Decision-MCTP, thereby completing the proof.

Consider an instance of the knapsack problem with $U = \{U_1, \ldots, U_n\}$ (i.e. $|U| = n$). We will construct an equivalent Decision-MCTP instance on the network graph depicted in figure 2.13. The graph consists of one node $X_i$ for each $u_i \in U$, $i \in \{1, \ldots, n\}$ and two other nodes $Y$ and $Z$. There is one edge from each $X_i$ to $Y$ and another edge from $Y$ to $Z$. Let the length of all these edges be 1, $N((X_i, Y)) = 1$ for every $i \in \{1, \ldots, n\}$ and $N((Y, Z)) = F + n$.

Let $\mathcal{R} = \{r_1, \ldots, r_n\}$, line $r_i$ being a line running from $X_i$ via $Y$ to $Z$ and back. Further, let $\mathcal{T}_{i,1}$ and $\mathcal{T}_{i,2}$ be the feasible line types for line $r_i$. Set $\mathcal{T} = \{\mathcal{T}_{1,1}, \mathcal{T}_{1,2}, \ldots, \mathcal{T}_{n,1}, \mathcal{T}_{n,2}\}$. For each train type, let 1 be the only feasible number of coaches. Moreover, let $\hat{\gamma}(r, \tau) = 1$ for each pair with $r \in \mathcal{R}$, $\tau \in \mathcal{T}$.

Knapsack problem:

| | | |
|---|---|---|
| Given: | $U$ | set |
| | $h : U \to \mathbb{N}$ | size function |
| | $f : U \to \mathbb{N}$ | value function |
| | $H$ | size limit |
| | $F$ | value bound |
| Find: | $U' \subseteq U$ such that | |
| | $\sum_{u \in U'} f(u) \geq F$ and $\sum_{u \in U'} h(u) \leq H$ | |
| | or state infeasibility | |

Figure 2.12: Knapsack problem

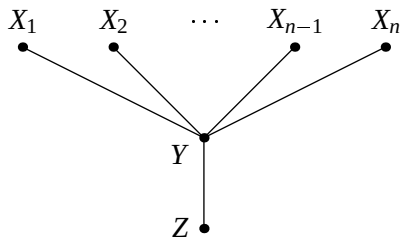



Figure 2.13: Network from the proof of theorem 2.4

Define the cost structure for the instance as follows: $C^{\text{fix}}(\tau) = C^{\text{km}}(\tau) = C^{\text{kmC}}(\tau) = 0$ for every $\tau \in \mathcal{T}$, $C^{\text{fixC}}(\mathcal{T}_{i,1}) = 1$ for every $i \in \{1,\ldots,n\}$, $C^{\text{fixC}}(\mathcal{T}_{i,1}) = h(u_i) + 1$ for every $i \in \{1,\ldots,n\}$. Let the capacity be $\mathfrak{C}(T_{i,1}) = 1$ and $\mathfrak{C}(T_{i,2}) = f(u_i) + 1$ for every $i \in \{1,\ldots,n\}$. Finally, define $K := H + n$.

Obviously, this Decision-MCTP instance can be constructed from the knapsack problem instance in polynomial time. It remains to show that the knapsack instance is feasible, if and only if the Decision-MCTP instance is feasible.

Let the knapsack instance be feasible, and let $U' \subseteq U$ be a set satisfying the knapsack problem constraints. In this case, we obtain a solution for the Decision-MCTP instance by setting:

$$x(r_i) = \begin{cases} \mathcal{T}_{i,1} \text{ if } u_i \notin U' \\ \mathcal{T}_{i,2} \text{ if } u_i \in U' \end{cases} \quad \text{and} \quad w(r_i) = 1$$

for each $i \in \{1,\ldots,n\}$. Of course, the first two conditions are satisfied by this choice, the same holds for the traveler capacity constraints on the edges $(X_i, Y)$. Now consider the capacity on the edge $(Y, Z)$:

$$\sum_{i=1}^{n} w(r_i) \cdot \mathfrak{C}(x(r_i)) = \sum_{i:\ u_i \in U'} (f(u_i) + 1) + \sum_{i:\ u_i \notin U'} 1 \geq F + n = N((Y,Z)).$$

The cost constraint is also fulfilled:

$$\sum_{i=1}^{n} w(r_i) \cdot C^{\text{fixC}}(x(r_i)) = \sum_{i:\ u_i \in U'} (h(u_i)+1) + \sum_{i:\ u_i \notin U'} 1 \leq H + n = K.$$

Conversely, let the Decision-MCTP instance be feasible and let $x$ and $w$ be given such that the corresponding conditions hold. By choosing

$$U' = \{u_i \mid x(r_i) = \mathcal{T}_{i,2},\ i = 1,\ldots,n\}$$

the value constraint of the knapsack problem is true because of

$$\begin{aligned} \sum_{i:\ x(r_i)=\mathcal{T}_{i,1}} 1 + \sum_{i:\ x(r_i)=\mathcal{T}_{i,2}} (f(u_i)+1) &\geq F + n \\ \sum_{i:\ x(r_i)=\mathcal{T}_{i,2}} f(u_i) &\geq F \\ \sum_{i:\ u_i \in U'} f(u_i) &\geq F \end{aligned}$$

Analogously, the size constraint can be verified. □

At this point, one could conjecture that algorithms for knapsack problems could be used to solve practical instances of the MCTP, but there is another difficulty:

**Theorem 2.5** *The Decision-MCTP is* NP*-complete even if there is only one train type.*

**Proof:** We formulate the Decision-MCTP with only one train type (*Decision-MCTP1*) as shown in figure 2.14. Because there is only one train type, the coach capacity can be scaled such that $\mathfrak{C} = 1$, and thus the capacity function is omitted for the Decision-MCTP1. Since the cost for motor units are constant if there is only one train type, those costs are not contained in the Decision-MCTP1.

Of course, the Decision-MCTP1 is in NP. We may modify every Decision-MCTP1 instance to an equivalent instance with feasible number of coaches between 0 and $\overline{W} - \underline{W}$ by setting (in this order:)

$$\begin{aligned} &N(e) := \max\left\{\left(N(e) - \sum_{r \in \mathcal{R},\ r \ni e} \underline{W}\right), 0\right\} \quad \text{for each } e \in E \\ &K := K - \sum_{r \in \mathcal{R}} \underline{W} \cdot (\hat{\gamma}(r) \cdot C^{\text{fixC}} + d(r) \cdot C^{\text{kmC}}) \\ &\overline{W} := \overline{W} - \underline{W} \\ &\underline{W} := 0 \end{aligned}$$

If this procedure leads to $K < 0$, we immediately know it is infeasible.

In the following, we show that every instance of the problem of *finding feasible line plans with frequency bound* 1, which is introduced and shown to be NP-complete in [10], can be polynomially transformed to such a modified Decision-MCTP1 instance. The problem of finding feasible line plans is the problem of choosing some lines from a given set of lines such that for each network edge the sum of the frequency of the lines running over it is bounded by certain numbers. In [10] it is shown that this problem is NP-complete even if for all edges, the lower and upper bounds are equal to 1.

Decision version of the MCTP with one train type (Decision-MCTP1):

| | | |
|---|---|---|
| Given: | $G = (V,E)$ | network graph |
| | $\mathcal{R}$ | set of lines |
| | $C^{\text{fixC}}, C^{\text{kmC}} \in \mathbb{N}_0$ | cost coefficients |
| | $\underline{W}, \overline{W} \in \mathbb{N}$ | bounds for numbers of coaches |
| | $d : \mathcal{R} \to \mathbb{N}$ | length of line circulation |
| | $\hat{\gamma} : \mathcal{R} \to \mathbb{N}$ | estimated number of train compositions |
| | $N : E \to \mathbb{N}$ | number of travelers |
| | $K \in \mathbb{N}$ | cost limit |
| Find: | $w : \mathcal{R} \to \mathbb{N}$ such that | |
| | 1. $\underline{W} \leq w(r) \leq \overline{W}$ for each $r \in \mathcal{R}$ | |
| | 2. $\sum_{r \in \mathcal{R}:\ r \ni e} w(r) \geq N(e)$ for each $e \in E$ | |
| | 3. $\sum_{r \in \mathcal{R}} \hat{\gamma}(r) \cdot w(r) \cdot C^{\text{fixC}} + d(r) \cdot w(r) \cdot C^{\text{kmC}} \leq K$ | |
| | or state infeasibility | |

Figure 2.14: Decision version of the MCTP with one train type

In this case, the only possible line frequency is 1. Therefore it is sufficient to show the polynomial transformation of instances of *finding feasible line plans with frequency bound* 1 *(FLP1)*, figure 2.15, to Decision-MCTP1 instances in order to prove that the Decision-MCTP1 is NP-complete.

For an FLP1 instance, we construct a corresponding Decision-MCTP1 instance by choosing $C^{\text{fixC}} = 0$, $C^{\text{kmC}} = 1$, $\underline{W} = 0$, $\overline{W} = 1$, $N(e) = 1$ for each $e \in E$, $d(r) =$ the number of edges in $r$ for each $r \in \mathcal{R}$, $\hat{\gamma}(r) = 1$ for each $r \in \mathcal{R}$, $K = |E|$. This is obviously a polynomial transformation. It remains to show that this instance is feasible if and only if the FLP1 instance is feasible.

Let the FLP1 instance be feasible with solution $\mathcal{R}' \subseteq \mathcal{R}$. Define

$$w(r) = \begin{cases} 1 \text{ if } r \in \mathcal{R}' \\ 0 \text{ if } r \notin \mathcal{R}'. \end{cases}$$

The capacity constraint is satisfied trivially. The cost constraint is also fulfilled:

$$\sum_{r \in \mathcal{R}} d(r) \cdot w(r) = \sum_{r \in \mathcal{R}'} d(r) = \sum_{r \in \mathcal{R}'} \sum_{e \in r} 1 = \sum_{e \in E} \sum_{r \in \mathcal{R}',\ r \ni e} 1 = |E| \leq K$$

Conversely, let the Decision-MCTP1 instance be feasible with a solution $w$. Choose $\mathcal{R}' := \{r \in \mathcal{R} \mid w(r) = 1\}$.

Consider an edge $e \in E$. From the capacity constraint of the Decision-MCTP1 instance we get

$$\sum_{r \in \mathcal{R}',\ r \ni e} 1 = \sum_{r \in \mathcal{R},\ r \ni e} w(r) \geq 1.$$

Feasible line plan problem with frequency bound 1 (FLP1):

Given: $G = (V, E)$ network graph

$\mathcal{R}$ set of lines

Find: $\mathcal{R}' \subseteq \mathcal{R}$ such that

$$\sum_{r \in \mathcal{R},\ r \ni e} 1 = 1 \text{ for each } e \in E$$

or state infeasibility

Figure 2.15: Feasible line plan problem with frequency bound 1

Now suppose that there is an edge $e' \in E$ with $\sum_{r \in \mathcal{R},\ r \ni e} 1 = \alpha > 1$. This would be a contradiction to the cost constraint of the Decision-MCTP1 instance, because

$$\sum_{r \in \mathcal{R}} d(r) \cdot w(r) = \sum_{r \in \mathcal{R}'} \sum_{e \in r} 1 = \sum_{e \in E} \sum_{\substack{r \in \mathcal{R}' \\ r \ni e}} 1 = \sum_{\substack{r \in \mathcal{R}' \\ r \ni e'}} 1 + \sum_{\substack{e \in E \\ e \neq e'}} \sum_{\substack{r \in \mathcal{R}' \\ r \ni e}} 1 \geq \alpha + (|E| - 1) > |E| = K.$$

This completes the proof. □

By the theorems of this section, we have seen that all principle parts of the cost optimal scheduling problem, i.e. the

- selection of train types (cf. theorem 2.4),
- selection of numbers of coaches (theorem 2.5),
- determination of a schedule for given train types, i.e. known intervals for travel time (theorem 2.3)

belong to a class of problems supposed to be difficult to solve. This motivates the use of a MIP model for the MCSP.

A direct solution of the MIPs of figure 2.10 for practical instances is impossible in a reasonable amount of time (i.e. even small practical instances required solution times of several days). For a strategic planning tool, the solution times should not exceed a few minutes.

In chapter 4, we will develop a decomposition method which accelerates the solution process significantly. With the method, it is possible to solve instances (or at least to get feasible solutions of acceptable quality) of practical interest in a few minutes.

# Chapter 3

# Feasible Schedules

In this chapter we discuss known algorithms and design new algorithms for solving PESP instances. The solution of PESP instances will form a crucial part of our schedule optimization algorithms which will be introduced in chapter 4.

### Notation and Concepts

Many PESP algorithms are based on ideas related to the event graph formulation of PESP (cf. section 2.3). We will shortly introduce some further notations and concepts before examining PESP algorithms.

The event graph $\mathcal{G} = (V_\mathcal{G}, A_\mathcal{G})$ was introduced in section 2.3. It can have parallel arcs. Let $n := |V_\mathcal{G}|$ and $m := |A_\mathcal{G}|$. In order to get a short notation, let the sets $V_\mathcal{G}$ and $A_\mathcal{G}$ be ordered and let the elements of $V_\mathcal{G}$ be called $1, \dots, n$. We will use the notation $a \in A_\mathcal{G}$, $a : i \to j$ to describe that $a$ is an arc from node $i \in V_\mathcal{G}$ to node $j \in V_\mathcal{G}$. $i$ and $j$ are called *endpoints* of $a$.

A *chain* $(a_1, \dots, a_r)$ is a sequence of arcs such that $a_i$ and $a_{i+1}$ $(1 \le i \le r-1)$ are adjacent, i.e. they have a common node. That endpoint of $a_1$ which is not an endpoint of $a_2$ and that endpoint of $a_r$ which is not an endpoint of $a_{r-1}$ are called *endpoints* of the chain. A chain is said to be *elementary* if, for each node which is an endpoint of an arc of the chain, there is at most one arc starting from and at most one arc ending in the node. A *cycle* is a chain whose endpoints coincide. If there is a mapping $\nu : \{1, \dots, r\} \to V_\mathcal{G}$ such that $a_i : \nu(i) \to \nu(i+1)$ for each $i \in \{1, \dots, r-1\}$, the chain is also called *path*. If, in addition to this condition, $a_r : \nu(r) \to \nu(1)$ for a cycle, the cycle is also called *circuit*.

Let $(a_1, \dots, a_r)$ be a chain with $a_i \neq a_j$ for each $1 \le i < j \le r$. Let there be no loop in $\{a_1, \dots, a_r\}$. An arc $a_\rho$, $\rho \in \{1, \dots, r-1\}$ is said to have *positive orientation* in the chain, if $a_\rho : i \to j$ and $j$ is an endpoint of $a_{\rho+1}$ (otherwise it is said to have negative orientation). $a_r$ is said to have positive orientation, if $a_r : i \to j$ and $i$ is an endpoint of $a_{r-1}$ (analogously negative orientation is defined). The *incidence vector* $\boldsymbol{p} \in \mathbb{R}^m$ representing the chain is defined by

$$p_a := \begin{cases} 1 & \text{if } a = a_\rho \text{ for a } \rho \in \{1, \dots, r\} \text{ with positive orientation} \\ -1 & \text{if } a = a_\rho \text{ for a } \rho \in \{1, \dots, r\} \text{ with negative orientation} \\ 0 & \text{if } a \text{ is not contained in the chain.} \end{cases} \qquad \text{for each } a \in A_\mathcal{G}$$

Let $\boldsymbol{p}^+ := (\max\{0, p_a\})_{a \in A_\mathcal{G}}$ and $\boldsymbol{p}^- := (\max\{0, -p_a\})_{a \in A_\mathcal{G}}$. Note that $\boldsymbol{p} = \boldsymbol{p}^+ - \boldsymbol{p}^-$.

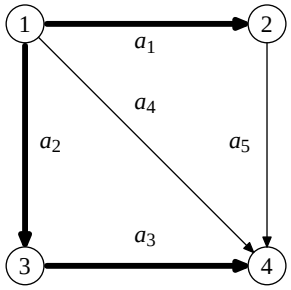
If, for every pair of nodes of $\mathcal{G}$, there is a chain with these nodes as endpoints, $\mathcal{G}$ is called *connected.* Every connected graph contains a *spanning tree* $\mathcal{T}$, which is a cycle-free, connected subgraph containing all nodes of $\mathcal{G}$. Let the arcs of $\mathcal{T}$ be denoted by $A_\mathcal{T}$. Let $\mu : A_\mathcal{G} \to \mathbb{R}$ be a cost function for the arcs of $\mathcal{G}$. $\mathcal{T}$ is called *minimum spanning tree* if the weight $\sum_{a \in A_\mathcal{T}} \mu(a)$ is minimal among the weights of all spanning trees. Calculating a minimum spanning tree can be efficiently done, for example, by the well known algorithms of Kruskal and Prim (see [1]). Arcs $a$ with $a \notin A_\mathcal{T}$ are called *non-tree*, *co-tree* arcs or *chords.*

The *node arc incidence matrix* $\mathfrak{N} = (\mathfrak{n}_{ia})$ has one row for each node $i$ and one column for each arc $a$. The entries of the matrix are defined by

$$\mathfrak{n}_{ia} := \begin{cases} 1 & \text{if } a : j \to i \text{ for some node } j \\ -1 & \text{if } a : i \to j \text{ for some node } j \\ 0 & \text{otherwise.} \end{cases}$$

An example for a graph and the corresponding node arc incidence matrix are given in figure 3.1. Let $\boldsymbol{\theta}_k$ denote the column of the *transposed* node arc incidence matrix corresponding to node $k$.

Adding a co-tree arc $a$ to a tree generates a unique elementary cycle. The incidence vector $\boldsymbol{\gamma}$ of this cycle with $\gamma_a = 1$ (note that if $\boldsymbol{\gamma}$ is an incidence vector of a cycle, $-\boldsymbol{\gamma}$ is the incidence vector of the same cycle, but with the direction reversed) forms a row of the *network matrix* $\Gamma$ of the graph. Since every spanning tree $\mathcal{T}$ has exactly $n-1$ arcs, there are $m-n+1$ co-tree arcs, and therefore $\Gamma$ has $m-n+1$ rows (and $m$ columns). The number $m-n+1$ is called *cyclomatic number* of the graph. When using a suitable numbering of the arcs, the network matrix can be split into the form $\Gamma = [N, E]$, where $E$ denotes the unit matrix associated with all co-tree arcs. An example for a such a network matrix is given in figure 3.1, where arcs represented by thick arrows form the spanning tree $\mathcal{T}$.




Node arc incidence matrix $\mathfrak{N}$

$$\begin{pmatrix} -1 & -1 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Network matrix $\Gamma$

$$\begin{pmatrix} 0 & -1 & -1 & 1 & 0 \\ 1 & -1 & -1 & 0 & 1 \end{pmatrix}$$

Figure 3.1: Graph and corresponding node arc incidence and network matrix

Let $\pi : V_\mathcal{G} \to \mathbb{R}$ be a potential (corresponding to a schedule) for $\mathcal{G}$. Let $\pi_i := \pi(i)$ for a node $i \in V_\mathcal{G}$, and let $\boldsymbol{\pi}$ be the vector of $\pi_i$, $i \in \{1, \ldots, n\}$. The corresponding tension, represented as a vector $\boldsymbol{x}$, can be calculated as $\boldsymbol{x} = \mathfrak{N}^T \boldsymbol{\pi}$. Every tension is characterized by the fact that the sum along a cycle is zero. This means $\boldsymbol{x}$ is a tension (associated with a potential $\boldsymbol{\pi}$) if and only if $\Gamma \boldsymbol{x} = \boldsymbol{0}$.

For an arc $a \in A_\mathcal{G}$, let $l_a$ be the lower and $u_a$ be the upper bound for the interval of the constraint corresponding to $a$. The interval $[l_a, u_a]$ is called *span*, and the expression $u_a - l_a$ is called *span length.*

In contrast to the span,

$$[l_a, u_a]_T := \{x - z \cdot T \mid x \in \mathbb{R},\ z \in \mathbb{Z}, l_a \leq x \leq u_a\}$$

denotes the so-called *periodic extension* of $[l_a, u_a]$. By this definition, (2.1) can directly be interpreted as being member of the corresponding set.

We will use the notation

$$(\cdot) \bmod T : \begin{cases} \mathbb{R} & \to & [0, T) \\ x & \mapsto & (x) \bmod T := \min\{x - z \cdot T \mid z \in \mathbb{Z},\ x - z \cdot T \geq 0\} \end{cases}$$

(it follows that $\pi_i \equiv \pi_j \bmod T$ is equivalent to $(\pi_i) \bmod T = (\pi_j) \bmod T$). This notation is extended to vectors $\boldsymbol{x}$ and sets $X$ by $(\boldsymbol{x}) \bmod T := ((x_i) \bmod T)$ and $X \bmod T := \{(x) \bmod T \mid x \in X\}$.

## 3.1 Preprocessing

The running times of the algorithms presented in this chapter exponentially depend on the size of the PESP event graph. Often the solution can be accelerated remarkably by reducing the graph size in a preprocessing step before actually starting the solution algorithm. The preprocessing methods discussed in this section can be divided into two categories:

- Reducing the number of nodes and arcs of the graph
- Reducing the interval width of the periodic interval constraints

### Reducing the Number of Nodes and Arcs of the Graph

There are some situations where nodes or arcs can be deleted from an instance without changing its feasibility status:

- *Trivially feasible or infeasible arcs*: If the graph contains an arc $a$ with $u_a - l_a \geq T$, the constraint corresponding to $a$ can obviously always be satisfied. Therefore, $a$ can be deleted from the graph.

  Similarly, if $u_a - l_a < 0$, the constraint cannot be satisfied at all, and the instance is infeasible.

  If there is a loop $a : i \to i$ with the interval $[l_a, u_a]$ containing a multiple of $T$, the arc can be deleted. If the interval does not contain such a multiple, the problem is infeasible.

- *Arcs with single point interval*: If there is an arc $a : i \to j$ with $l_a = u_a$, then the arc and one of the nodes can be deleted: Replace every arc $a' : i' \to j$ with interval $[l_{a'}, u_{a'}]$ by an arc $a'' : i' \to i$ with interval $[l_{a'} - l_a, u_{a'} - l_a]$. Analogously, arcs $a' : j \to i'$ for nodes $i'$ can be replaced. Now node $j$ and arc $a$ can be deleted.

- *Nodes with only one incident arc*: If there is a node $j$ which is only incident to one arc ($a : j \to i$ or $a : i \to j$), the constraint corresponding to $a$ can always be satisfied, and node $j$ and arc $a$ can be deleted.

- *Nodes with only two incident arcs*: If there is a node $j$ with only two incident arcs $a : i \to j$ and $a' : j \to k$, the node and the two arcs can be replaced by one arc $a'' : i \to k$ with interval $[l_a + l_{a'}, u_a + u_{a'}]$.

- *Components*: If the graph consists of several components (i.e. the underlying undirected graph is not connected), the PESP instances for the components can be solved separately. If and only if all these instances are feasible, the whole instance is feasible.

  If the graph consists of two parts that are only connected by one arc, the parts can be solved separately, and the potentials of the solution of one part have to be increased/decreased by a constant in order to get a feasible tension on the connecting arc.

  If the graph consists of two parts that are only connected by two arcs $a : i \to j$ and $a' : i' \to j'$ (without loss of generality it is assumed that $i$ and $i'$ are in the same part), we can choose one part (without loss of generality the part with $i$ and $i'$) and solve $T$ PESP instances arising from adding an arc $a'' : i \to i'$ with interval $[k,k]$, $k \in \{0,\dots,T-1\}$ to the part (assuming that only integer data is considered). By doing this, all feasible tensions $\pi_{i'} - \pi_i$ and thus all feasible tensions $\pi_{j'} - \pi_j$ can be determined. Arcs corresponding to the constraints for $\pi_{j'} - \pi_j$ can now be added to the part of the graph containing $j$ and $j'$. Now this part can be solved. This method is only useful if one part is "small" compared to the other one (since $T$ PESP instances for this part have to be solved), and if the number of constraints for $\pi_{j'} - \pi_j$ does not grow too much.

A feasible solution of the remaining problem(s) can obviously be extended to a solution of the original problem.

### Reducing Interval Widths

Let $a \in A_G$ with feasible interval $[l_a, u_a]$. Often it happens that for every feasible solution, the tension of arc $a$ is an element of a subset $S \subsetneq [l_a, u_a]$. Sometimes, the interval may then be replaced by another interval $S' := [l'_a, u'_a]$ with $S \subseteq S' \subsetneq [l_a, u_a]$. Sometimes, it is possible to detect such a possibility in a preprocessing step.

In order to develop such a preprocessing method, we will use a constraint propagation approach: Look at the example of figure 3.2. There, the time span $[20,30]_{60}$ is given for the arc from node 1 to node 2. However, $\pi_2 - \pi_1 \equiv 20 \bmod 60$ is not possible because of the other two arcs. We can actually replace the span $[20,30]_{60}$ by $[21,30]_{60}$. The constraint propagation method investigates all triples of nodes in a recursive manner, until no intervals can be reduced. We will now give a formal description of this approach.

Let $T$ denote a fixed period. A set $U \subseteq \mathbb{R}$ is said to be *$T$-periodic*, if for all $u \in U$ and $z \in \mathbb{Z}$ also $u + zT \in U$. Such a set can be written as

$$U := \{u + zT \mid u \in \hat{U} \subseteq [0,T),\ z \in \mathbb{Z}\}.$$

Let $S_T := \{s + zT \mid s \in S,\ z \in \mathbb{Z}\}$. $S_T$ is a periodic set. If $a : i \to j$ is an arc, then $[l_a, u_a]_T$ is the periodic set containing the feasible tension values $\pi_j - \pi_i$. Let the set of all $T$-periodic sets be denoted by *Per* and define the operations

$$U \oplus V := U \cap V \qquad \text{and} \qquad U \otimes V := \{u + v \mid u \in U, v \in V\}$$

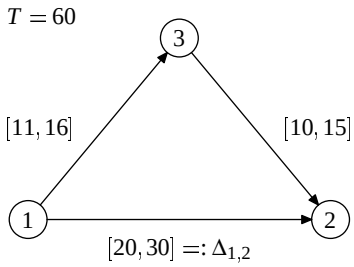



Figure 3.2: The span $\Delta_{1,2}$ can be replaced by $[21,30]$

for $U, V \in Per$.

Algorithm 3.1 calculates a periodic set $M_{ij}$ for each pair of nodes $(i, j)$. Each feasible potential $\pi$ has to satisfy $\pi_j - \pi_i \in M_{ij}$ for every arc $a : i \to j$. If $0 \neq M_{ii}$ for a node $i$, then the instance is infeasible, because no potential can fulfill $\pi_i - \pi_i \not\equiv 0 \bmod T$.

In contrast to this, $0 \in M_{ii}$ for all nodes does not imply that the PESP instance is feasible, as one can see from the counter example in figure 3.3.

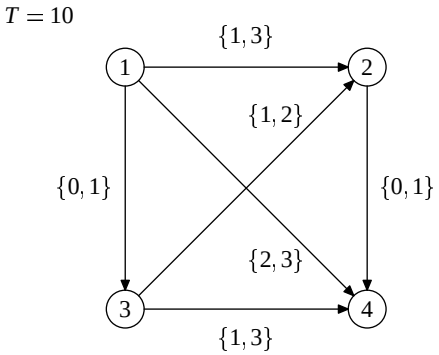



Figure 3.3: Infeasible PESP instance with $0 \in M_{ii}$ for each node $i$

In the example, non-convex spans are given. However, these can be modeled by intersections of several interval constraints (cf. section 2.1).

On the one hand, the PESP instance is infeasible: Assume $\pi_1 \equiv 0 \bmod 10$, then there are two cases: $\pi_4 \equiv 2 \bmod 10 \Rightarrow \pi_2 \equiv 1 \bmod 10 \Rightarrow \pi_3 \equiv 0 \bmod 10 \Rightarrow \pi_4 - \pi_3 \notin \{1,3\}_T$, $\pi_4 \equiv 3 \bmod 10 \Rightarrow \pi_2 \equiv 3 \bmod 10 \Rightarrow \pi_3 \equiv 1 \bmod 10 \Rightarrow \pi_4 - \pi_3 \notin \{1,3\}_T$.

On the other hand, for each arc $a : i \to j$, the value of $M_{i,j}$ is given by the original span of figure 3.3, and $0 \in M_{i,i}$ for each node $i$.

Our preprocessing method will now work as follows: Let $a : i \to j$ be an arc with interval $[l_a, u_a]_T$. For every feasible solution of the PESP instance, $\pi_j - \pi_i \in [l_a, u_a]_T \cap M_{ij}$. We can construct a stronger initial constraint system in this way:

- We can replace the arc $a$ by several arcs in such a way that $\pi_j - \pi_i \in [l_a, u_a]_T \cap M_{ij}$ is demanded explicitely (recall that $M_{ij}$ may be a union of periodic intervals). By this procedure, the number of arcs may be increased to such an extent that the PESP algorithm becomes very slow.

**Algorithm 3.1** Constraint Propagation for Preprocessing

**for** each $(i,j) \in V_{\mathcal{G}} \times V_{\mathcal{G}}$ **do**
  **if** $i = j$ **then**
    $M_{ij} := T \cdot \mathbb{Z}$
  **else if** there is an arc $a : i \to j$ or $a : j \to i$ **then**
    $M_{ij} := \left(\bigcap_{a:i\to j}[l_a, u_a]_T\right) \cap \left(\bigcap_{a:j\to i}[-u_a, -l_a]_T\right)$
  **else**
    $M_{ij} := [0,T)_T$
  **end if**
**end for**
*modification* := **true**
**while** *modification* **do**
  *modification* := **false**
  **for** all $(i,j,k) \in V_{\mathcal{G}} \times V_{\mathcal{G}} \times V_{\mathcal{G}}$ with $k \neq i,j$ **do**
    $A := M_{ik} \otimes M_{kj}$
    **if** $i = j$ **and** $0 \notin M_{ij}$ **then**
      Stop. Instance is infeasible
    **end if**
    **if** $A \not\supseteq M_{ij}$ **then**
      $M_{ij} := M_{ij} \oplus A$
      *modification* := **true**
    **end if**
  **end for**
**end while**
Stop. $M_{ij}$ has been calculated for all pairs of nodes.

- We only modify the interval bounds $l_a$ and $u_a$ for the arc in such a way that $u_a - l_a$ is minimized, but still $M_{ij} \subseteq [l_a, u_a]_T$ is fulfilled. An example is given in figure 3.4.

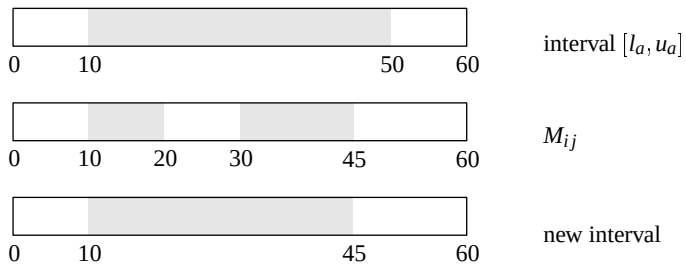



Figure 3.4: Reducing the interval width

## 3.2 Basic Properties of the PESP

Recall that a PESP instance is given by a time period $T$, a graph $\mathcal{G} = (V_\mathcal{G}, A_\mathcal{G})$, $n := |V_\mathcal{G}|$, $m := |A_\mathcal{G}|$, a set of spans $\{[l_a, u_a] \mid a \in A_\mathcal{G}\}$, and to solve the PESP instance means finding a potential $\boldsymbol{\pi} \in \mathbb{R}^n$ and a vector of modulo parameters $\boldsymbol{z} \in \mathbb{Z}^m$ with

$$l_a \le \pi_j - \pi_i - z_a \cdot T \le u_a \qquad \text{for each } a : i \to j \in A_\mathcal{G}. \tag{3.1}$$

We will shorten our notation by $V := V_\mathcal{G}$ and $A := A_\mathcal{G}$.

Let the set of feasible solutions for a PESP instance be denoted by

$$Q := \{(\boldsymbol{\pi}, \boldsymbol{z}) \mid \boldsymbol{\pi} \in \mathbb{R}^n, \boldsymbol{z} \in \mathbb{Z}^m, l_a \le \pi_j - \pi_i - z_a \cdot T \le u_a \text{ for each } a : i \to j \in A\}$$

The convex hull $\operatorname{conv} Q$ of this set is called the *(unbounded) timetable polyhedron.* For a fixed vector of modulo parameters $\boldsymbol{z} \in \mathbb{Z}^m$, define

$$\Pi(\boldsymbol{z}) := \{\boldsymbol{\pi} \mid l_a \le \pi_j - \pi_i - z_a \cdot T \le u_a \text{ for each } a : i \to j \in A\},$$

and let

$$\mathcal{Z} := \{\boldsymbol{z} \in \mathbb{Z}^m \mid \Pi(\boldsymbol{z}) \ne \emptyset\}.$$

The problem of deciding

$$\Pi(\boldsymbol{z}) \stackrel{?}{=} \emptyset$$

for a given $\boldsymbol{z}$ is a feasible differential problem (see appendix C.3) with spans $[\boldsymbol{l} + \boldsymbol{z} \cdot T, \boldsymbol{u} + \boldsymbol{z} \cdot T]$ and can be solved by a shortest path problem in a modified graph $\mathcal{G}'$ with $\mathcal{G}' = (V_\mathcal{G}, A_+ \cup A_-)$, $A_+ := A$, $A_- :=$ the set of counter arcs for each $a \in A$ (cf. appendix C.3). The arc lengths for $\mathcal{G}'$ are given by

$$\mu_a := \begin{cases} u_a + z_a \cdot T & a \in A_+ \\ -l_a - z_a \cdot T & a \in A_-. \end{cases}$$

According to (C.2), the feasible differential problem is soluble if and only if for each cycle with incidence vector $\boldsymbol{\gamma}$, $\boldsymbol{\mu}^T(\boldsymbol{\gamma}^+ + \boldsymbol{\gamma}^-) \ge 0$. From

$$\boldsymbol{\mu}^T(\boldsymbol{\gamma}^+ + \boldsymbol{\gamma}^-) = (\boldsymbol{u} + \boldsymbol{z} \cdot T)^T \boldsymbol{\gamma}^+ - (\boldsymbol{l} + \boldsymbol{z} \cdot T)^T \boldsymbol{\gamma}^- = \boldsymbol{u}^T \boldsymbol{\gamma}^+ - \boldsymbol{l}^T \boldsymbol{\gamma}^- + T \boldsymbol{z}^T \boldsymbol{\gamma}$$

it follows that $\Pi(\boldsymbol{z}) \ne \emptyset$ is equivalent to

$$\boldsymbol{u}^T \boldsymbol{\gamma}^+ - \boldsymbol{l}^T \boldsymbol{\gamma}^- + T \boldsymbol{z}^T \boldsymbol{\gamma} \ge 0 \Leftrightarrow \boldsymbol{z}^T \boldsymbol{\gamma} \ge \frac{1}{T} \cdot (\boldsymbol{l}^T \boldsymbol{\gamma}^- - \boldsymbol{u}^T \boldsymbol{\gamma}^+). \tag{3.2}$$

Since $\boldsymbol{z}$ is integral, we have the following proposition:

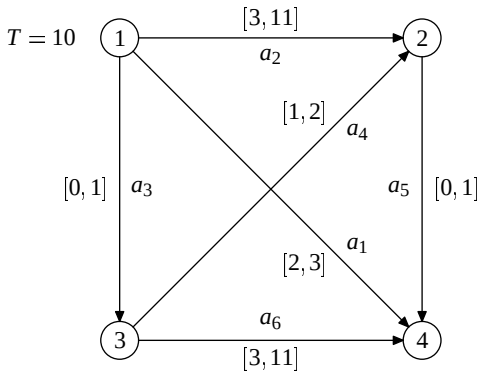
**Proposition 3.1** *$\Pi(\boldsymbol{z}) \ne \emptyset$ if and only if for every elementary cycle*

$$\boldsymbol{\gamma}^T \boldsymbol{z} \ge \left\lceil \frac{1}{T} \left( \boldsymbol{l}^T \boldsymbol{\gamma}^- - \boldsymbol{u}^T \boldsymbol{\gamma}^+ \right) \right\rceil. \tag{3.3}$$

This result has also been shown by polyhedral arguments in [47]. The inequalites of this proposition can be used as cutting planes and will play an important role in this chapter. They are called *cycle cutting planes.*

The cycle cutting planes cannot be used in order to remove all non-integer solutions from the inequality system (3.1). Figure 3.5 shows an example. The constraint system consisting of the inequalities (3.1) and all the cycle cutting planes has the fractional solution $\boldsymbol{\pi} = (0,7,1,2)^T$, $\boldsymbol{z} = (0,0,0,\frac{1}{2},-\frac{1}{2},-\frac{1}{2})^T$. The PESP instance is infeasible: Let $\pi_1 \equiv 0 \bmod 10$. Then we have to consider the following two cases: $\pi_3 \equiv 0 \bmod 10$ and $\pi_3 \equiv 1 \bmod 10$. Proposition 3.4 will show that if there is no integral solution to a PESP instance with integer data $\boldsymbol{l}$, $\boldsymbol{u}$ and $T$, then there is no solution at all.

$\pi_3 \equiv 0 \bmod 10 \Rightarrow \pi_2 \equiv 1 \bmod 10$ because of arc $a_4$. From arc $a_1$ and arc $a_5$ it follows that $\pi_4 \equiv 2 \bmod 10$, which is a contradiction to arc $a_6$. One can obtain an analogous contradiction for $\pi_3 \equiv 1 \bmod 10$.




Cycle cutting planes :

$$
\begin{array}{rcccl}
-1 & \leq & z_2 - z_3 - z_4 & \leq & 0 \\
0 & \leq & z_1 - z_2 - z_5 & \leq & 1 \\
0 & \leq & z_4 + z_5 - z_6 & \leq & 1 \\
0 & \leq & z_1 - z_3 - z_6 & \leq & 1 \\
-1 & \leq & -z_1 + z_2 - z_4 + z_6 & \leq & -1 \\
0 & \leq & z_2 - z_3 + z_5 - z_6 & \leq & 0 \\
0 & \leq & z_1 - z_3 - z_4 - z_5 & \leq & 0
\end{array}
$$

Figure 3.5: Cycle cutting planes for a PESP instance

The following result has been proven in [49] and [59]:

**Proposition 3.2** *If a PESP instance is feasible, then for each vector of modulo parameters $\boldsymbol{z}'$ and for each fixed spanning tree there exists a vector of modulo parameters $\boldsymbol{z}$ with $z_a = 0$ for all tree arcs and*

$$\Pi(\boldsymbol{z}') \bmod T = \Pi(\boldsymbol{z}) \bmod T \neq \emptyset.$$

**Proof:** Consider a fixed spanning tree of the connected graph $\mathcal{G}$. Fix an arbitrary node, say node 1, as the tree root node. Then for each other node $i$, the tree contains a uniquely determined chain with incidence vector $\boldsymbol{p}^i$ from node 1 to node $i$. Let $\boldsymbol{z}' \in \mathbb{Z}^m$ and $\boldsymbol{\pi}' \in \Pi(\boldsymbol{z}')$.

Now define $\boldsymbol{\pi}$ and $\boldsymbol{z}$ by

$$\pi_k := \pi'_k - (\boldsymbol{p}^k)^T \boldsymbol{z}' \cdot T \qquad \text{and} \qquad z_a := z'_a - (\boldsymbol{p}^j - \boldsymbol{p}^i)^T \boldsymbol{z}'$$

for each node $k$ and each arc $a : i \to j$. We will now show that $z_a = 0$ for tree arcs and $\boldsymbol{\pi} \in \Pi(\boldsymbol{z})$.

Consider a tree arc $a : i \to j$. At first observe that $\boldsymbol{p}^j - \boldsymbol{p}^i = \boldsymbol{e}^a$, where $\boldsymbol{e}^a$ denotes the unit vector with $e^a_a = 1$ and $e^a_b$ for each arc $b \neq a$. It follows that $z_a = 0$ and

$$\pi_j - \pi_i - z_a \cdot T = \pi'_j - (\boldsymbol{p}^j)^T \boldsymbol{z}' \cdot T - \pi'_i + (\boldsymbol{p}^i)^T \boldsymbol{z}' \cdot T = \pi'_j - \pi'_i - z'_a \cdot T \in [l_a, u_a].$$

For a co-tree arc $a : i \to j$, we have

$$\begin{aligned}\pi_j - \pi_i - z_a \cdot T &= \pi'_j - (\boldsymbol{p}^j)^T \boldsymbol{z}' \cdot T - \pi'_i + (\boldsymbol{p}^i)^T \boldsymbol{z} \cdot T + (-z'_a + (\boldsymbol{p}^j)^T \boldsymbol{z}' - (\boldsymbol{p}^i)^T \boldsymbol{z}') \cdot T \\ &= \pi'_j - \pi'_i - z'_a \cdot T \in [l_a, u_a].\end{aligned}$$

It follows that $\boldsymbol{\pi} \in \Pi(\boldsymbol{z})$. Since $\boldsymbol{\pi}' \equiv \boldsymbol{\pi} \bmod T$, the proof is complete. □

Now assume that we have a fixed spanning tree $\mathcal{T}$ with associated network matrix $\Gamma$. Two schedules given by $\boldsymbol{\pi}$ and $\boldsymbol{\pi}'$ with $\boldsymbol{\pi} \equiv \boldsymbol{\pi}' \bmod T$ are equivalent. Proposition 3.2 allows us to consider only schedules with modulo parameter of 0 on all tree arcs. Define

$$\mathcal{Z}_{\mathcal{T}} := \{\boldsymbol{z} \in \mathcal{Z} \mid z_a = 0 \text{ for } a \in \mathcal{T}\}.$$

Note that each row of $\Gamma$ is the (transposed) incidence vector $\boldsymbol{\gamma}_a^T$ of a cycle which contains only tree arcs and exactly one chord $a$. The induced cycle cutting planes for the cycle and its counter cycle give bounds on the modulo parameter of all chords by the following proposition:

**Proposition 3.3** *Let $\boldsymbol{\gamma}_a^T$ be a row of the network matrix $\Gamma$ and let $\boldsymbol{z} \in \mathcal{Z}_{\mathcal{T}}$. Then*

$$\underline{z}_a := \left\lceil \frac{1}{T}\left(\boldsymbol{l}^T \boldsymbol{\gamma}_a^- - \boldsymbol{u}^T \boldsymbol{\gamma}_a^+\right) \right\rceil \le z_a \le \left\lfloor \frac{1}{T}\left(\boldsymbol{u}^T \boldsymbol{\gamma}_a^- - \boldsymbol{l}^T \boldsymbol{\gamma}_a^+\right) \right\rfloor =: \overline{z}_a.$$

It follows that $\mathcal{Z}_{\mathcal{T}}$ is finite.

The next proposition deals with integral solutions of a PESP instance. It has, for example, been proven in [45].

**Proposition 3.4** *Let a feasible PESP instance be given by $V_{\mathcal{G}}$, $A_{\mathcal{G}}$, $\boldsymbol{l}, \boldsymbol{u} \in \mathbb{Z}^{|A_{\mathcal{G}}|}$ and $T \in \mathbb{Z}$. Then there exists a feasible potential $\boldsymbol{\pi} \in \mathbb{Z}^{|V_{\mathcal{G}}|}$.*

**Proof:** Since the PESP instance is feasible, there is a vector $\boldsymbol{z} \in \mathbb{Z}^{|A_{\mathcal{G}}|}$ such that $\boldsymbol{l} \le \mathfrak{N}^T \boldsymbol{\pi} - \boldsymbol{z}T \le \boldsymbol{u}$. This constraint system can also be written as

$$\begin{pmatrix} \mathfrak{N}^T \\ -\mathfrak{N}^T \end{pmatrix} \boldsymbol{\pi} \le \begin{pmatrix} \boldsymbol{u} + \boldsymbol{z}T \\ -\boldsymbol{l} - \boldsymbol{z}T \end{pmatrix}.$$

The coefficient matrix of this system is totally unimodular, and the right hand side is integral. Therefore an integer solution $\boldsymbol{\pi}$ exists. □

## 3.3 Mixed Integer Programming

A straightforward approach to solve PESP instances is the use of the mixed integer programming formulation given by (2.11). We can strengthen the formulation (i.e. add constraints such that the original LP solution gets infeasible, see also appendix B) in the following ways:

- The integer variables corresponding to the arcs of a spanning tree can be fixed to zero (proposition 3.2).
- Bounds for the other integer variables are obtained from proposition 3.3.
- Cycle cutting planes (3.3) may be used.

Additionally, we may fix the potential of one node to an arbitrary value (say $\pi_1 := 0$): If $\boldsymbol{\pi}$ is a feasible schedule, then also $\boldsymbol{\pi} + c \cdot \mathbf{1}$ with $c \in \mathbb{R}$ is a feasible schedule. $\mathbf{1}$ denotes the vector containing only 1-entries.

Our practical experience (see chapter 5) shows that the MIP solution process is possible for some instances. Nevertheless there is a problem with this approach: As we have already mentioned, in case of an infeasible instance, we would like to detect a "reason" for the infeasibility, or in other words, we would like to have an idea how to relax the instance in such a way that it becomes feasible. There is no obvious way for getting such information from the MIP branch-and-bound tree.

## 3.4 Odijk's Algorithm

In [47, 48], Odijk suggests a PESP algorithm based on the MIP formulation of PESP (cf. (2.10)). However, the algorithm does not solve the MIP instance directly, but in a two-step iterative procedure that profits from the effect of the cycle cutting planes (3.3). We will now describe some ideas leading to the algorithm. For more information, [47, 48] can be consulted.

Note that solving (2.10) for fixed $\boldsymbol{z}$, or equivalently finding a $\boldsymbol{\pi} \in \Pi(\boldsymbol{z})$, can be formulated as a linear programming problem and thus can be solved efficiently. Let $\boldsymbol{z} \in \mathbb{Z}^m$ and define $\boldsymbol{l}^{(\boldsymbol{z})} := \boldsymbol{l} + \boldsymbol{z} \cdot T$, $\boldsymbol{u}^{(\boldsymbol{z})} := \boldsymbol{u} + \boldsymbol{z} \cdot T$ and let

$$LP(\boldsymbol{z}) := \left\{ \begin{array}{ll} \max & 0 \\ \text{subject to} & \boldsymbol{l}^{(\boldsymbol{z})} \leq \mathfrak{N}^T \boldsymbol{\pi} \leq \boldsymbol{u}^{(\boldsymbol{z})} \\ & \boldsymbol{\pi} \geq 0 \end{array} \right. \tag{3.4}$$

$\boldsymbol{\pi} \geq 0$ does not really present a constraint for the schedule. The dual problem of $LP(\boldsymbol{z})$ is given by

$$DP(\boldsymbol{z}) := \left\{ \begin{array}{ll} \min & (\boldsymbol{u}^{(\boldsymbol{z})})^T \boldsymbol{y}_+ - (\boldsymbol{l}^{(\boldsymbol{z})})^T \boldsymbol{y}_- \\ \text{subject to} & \mathfrak{N}(\boldsymbol{y}_+ - \boldsymbol{y}_-) \geq 0 \end{array} \right. \tag{3.5}$$

This problem either has an optimal solution with objective value 0 (e.g. $\boldsymbol{y}_+ = 0$ and $\boldsymbol{y}_- = 0$) or is unbounded from below. In the first case, $LP(\boldsymbol{z})$ also has feasible solutions, and a schedule $\boldsymbol{\pi}$ can be determined. Otherwise, an extreme ray $(\boldsymbol{y}_+, \boldsymbol{y}_-)$ with $(\boldsymbol{u}^{(\boldsymbol{z})})^T \boldsymbol{y}_+ - (\boldsymbol{l}^{(\boldsymbol{z})})^T \boldsymbol{y}_- < 0$ can be found. In [47], Odijk shows that from this ray, a cycle cutting plane can be constructed which is violated by $\boldsymbol{z}$.

These ideas are integrated in the following iterative procedure: During each iteration, a polytope $P$ containing candidates for vectors of modulo parameters $\boldsymbol{z}$ is kept. By the help of a backtracking procedure, a $\boldsymbol{z} \in P \cap \mathbb{Z}^m$ is selected (this is the main time consuming part of the algorithm). If no such vector exists, the PESP instance is infeasible. Otherwise, $DP(\boldsymbol{z})$ is solved. If it is unbounded from below, a cycle cutting plane violated by $\boldsymbol{z}$ is constructed, and $P$ is replaced by the intersection of $P$

with the cut. If $DP(\boldsymbol{z})$ has an optimal solution with objective value 0, the iteration procedure can be stopped, since a solution $\pi$ of the PESP instance can be found.

Initially, $P$ is the polytope described by the bound constraints from proposition 3.3 and $z_a = 0$ for tree arcs for a fixed spanning tree $\mathcal{T}$.

The complete method is given by algorithm 3.2. Practical experiences show that this method can only handle small PESP instances in a reasonable amount of time (cf. [45], [47]).

**Algorithm 3.2** Odijk's Algorithm

Choose a spanning tree $\mathcal{T}$.
$P := \{\boldsymbol{z} \in \mathbb{R}^m \mid z_a = 0 \text{ for each } a \in A(\mathcal{T}) \text{ and } \underline{z}_a \leq z_a \leq \overline{z}_a \text{ for each } a \notin A(\mathcal{T})\}$
**loop**
  **if** $P \cap \mathbb{Z}^m = \emptyset$ **then**
    Stop. The PESP instance is infeasible.
  **end if**
  Choose $\boldsymbol{z} \in P \cap \mathbb{Z}^m$.
  **if** $DP(\boldsymbol{z})$ has an optimal solution $(\boldsymbol{y}_+, \boldsymbol{y}_-)$ with objective value 0 **then**
    Construct optimal solution $\boldsymbol{\pi}$ for $LP(\boldsymbol{z})$.
    Stop. $(\boldsymbol{\pi}, \boldsymbol{z})$ is a solution for the PESP instance.
  **end if**
  From an extreme ray $(\boldsymbol{y}_+, \boldsymbol{y}_-)$ with negative objective value for $DP(\boldsymbol{z})$, construct a cycle cutting plane $\boldsymbol{\alpha}^T \boldsymbol{z} \leq \alpha_0$ which is violated by $\boldsymbol{z}$.
  $P := P \cap \{\boldsymbol{z} \in \mathbb{R}^m \mid \boldsymbol{\alpha}^T \boldsymbol{z} \leq \alpha_0\}$
**end loop**

## 3.5 Constraint Propagation

Voorhoeve has developed another solution method for PESP instances in [62]. His algorithm extends the constraint propagation method introduced in section 3.1.

Assume that the potential of an arbitrary node has been fixed to an arbitrary value (e.g. $\pi_1 := 0$) and that $M_{ij}$ has been calculated for each pair $(i, j)$ of nodes. Consider these two cases:

- $0 \notin M_{ii}$ for some node $i$. Then the PESP instance is infeasible.
- $0 \in M_{ii}$ for all nodes $i$. In this case, the PESP instance *may* be feasible.

The calculation of $M_{ij}$ is integrated into the following constraint propagation procedure: If $0 \in M_{ii}$ for all nodes $i$, another potential, say $\pi_2$, is fixed in such a way that $\pi_2 - \pi_1 \in M_{12}$. This probably reduces the sets $M_{ij}$ for other pairs of nodes $(i, j)$. The procedure is repeated. If $0 \notin M_{ii}$ for a node $i$ at some step, one has to backtrack, and the potential of the variable that was fixed in the step before is given another value.

The algorithm terminates either with a feasible fixing $\boldsymbol{\pi}$ of all potentials or with a proof of infeasibility (all potentials lead to a backtracking step).

Voorhoeve's method is given as algorithm 3.3. Computational results with this method are rather deterring [4]. A main reason for the behavior is that there are too many possibilities for the fixing of potentials when $T$ is large (like $T = 60$). In this case, the search tree soon gets too large to be manageable.

**Algorithm 3.3** Voorhoeve's Constraint Propagation Algorithm

Choose an arbitrary node $i \in V$ and set $\pi_i := 0$.
**loop**
  Calculate $M_{ij}$ for each $(i, j) \in V \times V$.
  **if** $0 \notin M_{ii}$ for some node $i$ **then**
    **if** no backtracking is possible **then**
      Stop. The PESP instance is infeasible.
    **end if**
    Perform a backtracking step, i.e. change the potential of the variable that was fixed before. If necessary, do further backtracking. If no such backtracking is possible, stop. The PESP instance is infeasible.
  **else**
    Choose a node $i \in V$ such that $\pi_i$ is not fixed yet.
    Fix $\pi_i$ in such a way that $\pi_j - \pi_i \in M_{ij}$ for all nodes $j$.
  **end if**
**end loop**

## 3.6 Algorithm of Serafini and Ukovich

Serafini and Ukovich introduce a backtracking method for solving PESP instances in [59]. In this method, the possible vectors of modulo parameters are investigated.

The algorithm starts with determining a minimum spanning tree $\mathcal{T}$ concerning the span lengths $u_a - l_a$ for each $a \in A$ and a feasible potential $\boldsymbol{\pi}$ for the graph ignoring the chords. $\boldsymbol{\pi}$ is obtained by fixing the potential of one node and choosing an arbitrary feasible tension on the tree arcs. Afterwards, all chords are sorted in order of increasing span length. Let this order be $a_1, \ldots, a_{m-n+1}$.

Now assume that the algorithm is *searching at level k*, which means that

- The modulo parameters for the tree arcs are set to 0, and the modulo parameters for all chords $a_r$ with $r \in \{1, \ldots, k-1\}$ have been fixed.
- A potential $\boldsymbol{\pi}$ is known which is feasible up to level $k-1$, i.e. the periodic interval constraints (3.1) are satisfied for the tree arcs and for $a_1, \ldots, a_{k-1}$.
- The algorithm is looking for an integer $z_{a_k}$ for arc $a_k : i \to j$ such that $\boldsymbol{\pi}$ can be made feasible up to level $k$ without changing fixed modulo parameters.

The search produces one of these two results:

- There exists a $z_{a_k} \in \mathbb{Z}$ with $\pi_j - \pi_i - z_{a_k} \in [l_{a_k}, u_{a_k}]$. Therefore $\boldsymbol{\pi}$ is feasible up to level $k$.
- There is a $z \in \mathbb{Z}$ with

$$\pi_j - \pi_i - z \cdot T < l_{a_k} \leq u_{a_k} < \pi_j - \pi_i - (z-1) \cdot T.$$

In this case, the algorithm tries to raise or lower the tension value $\pi_j - \pi_i$ in such a way that the tension gets feasible for arc $a_k$ (cf. figure 3.6) and the feasibility for the other tensions is maintained (without changing any of the fixed modulo parameters). This is a feasible differential problem and can be solved by a modified Dijkstra shortest path procedure that is given in appendix C.

Suppose that one (or both) attempts, i.e. to lower or to raise the tension, fails. In this situation, the shortest path algorithm finds a circuit of negative length (cf. appendix C). The arcs of this circuit (these circuits) are called *blocking arcs* and play an important role for the backtracking step of the algorithm: The tension value of arc $a_k$ can only be lowered (or raised) for the required amount, if at least the modulo parameter of one of the blocking arcs is changed.

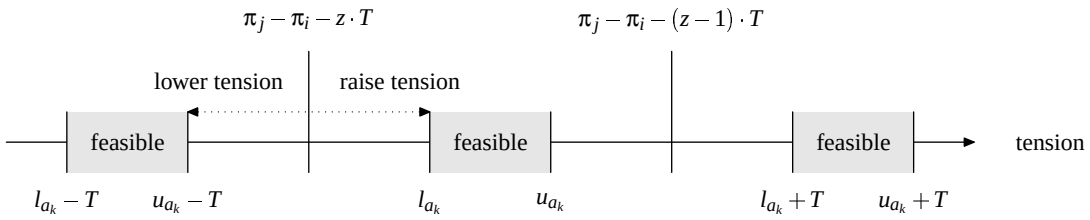



Figure 3.6: Raising or lowering the tension

The result of this proceeding is either a feasible modulo parameter $z_{a_k}$ and a (possibly modified) potential which is feasible up to level $k$ or infeasibility. Stating infeasibility means that it is not possible to extend the partial solution $\{z_{a_1}, \ldots, z_{a_{k-1}}\}$ to a feasible $\boldsymbol{z}$-vector for the complete instance (i.e. $\Pi(\boldsymbol{z}) = \emptyset$ for all $\boldsymbol{z} \in \mathbb{Z}^m$ with $\{z_{a_1}, \ldots, z_{a_{k-1}}\}$ fixed as done by the algorithm). In this case, the modulo parameter of a previously investigated level has to be changed. To be more exact, let $\mathcal{B}$ be the union of the sets of blocking arcs for lowering or raising the tension of arc $a_k$. Then the algorithm backtracks to level $k'$ with $k' := \max\{\kappa \mid a_\kappa \in \mathcal{B}\}$.

For each level $k$, information has to be stored on the values for $z_{a_k}$ that have already been tested. If all values $z_{a_k} \in \{\underline{z}_{a_k}, \ldots \overline{z}_{a_k}\}$ have lead to infeasibility, the algorithm has to backtrack to level $k-1$. In [59], Serafini and Ukovich suggest a method for storing all relevant information in a list structure (rather than a tree structure). However, their pseudo code contains an error. There, on page 565, line 7, the assignment $\mathcal{B} \leftarrow \emptyset$ may cause that parts of the search space are not investigated and infeasibility is stated although a feasible solution exists. Nevertheless, this error can be corrected as suggested by Nachtigall in [43].

The size of the backtracking search tree may be of order $\prod_{k=1}^{m-n+1} 1 + \overline{z}_{a_k} - \underline{z}_{a_k}$, i.e. exponential in the number of arcs. This explains why Serafini and Ukovich suggest to take a minimum spanning tree concerning span lengths an to order the arcs in order of increasing span lengths. One can heuristically

assume that the number of possible modulo parameters for level $k$ is smaller, as "more restrictive" the constraints of the start tree and the levels $1,\ldots,k-1$ are.

Besides computation time, there is another reason for keeping the search tree "as small as possible" motivated from our cost optimization algorithms in chapter 4: In case of an infeasible PESP instance, we will have to analyze the "reason" for the infeasibility. Formally, we will need to determine a set of arcs whose interval constraints cannot be satisfied simultaneously. It will be of advantage to find a small set of arcs with this property, and thus we would like to detect infeasibility of a PESP instance as soon as possible. Obviously, a set of arcs whose constraints cannot be satisfied simultaneously is contained in the set of arcs from the spanning tree and those chords that have been examined for modulo parameter fixing.

### Developments of Schrijver and Steenbeek

Schrijver and Steenbeek observe that the PESP algorithm of Serafini and Ukovich investigates some parts of the search space again and again [56]. This is caused by the fact that after a backtracking step from level $k$ down to level $k' < k$, the feasible modulo parameters for all levels $k'+1, k'+2, \ldots, k-1$ are forgotten by the algorithm and have to be recalculated by shortest path algorithms, which may be time consuming for larger graphs. The idea in [56] is to dynamically reorganize the search tree in such a way that this information can be kept after each backtracking step. In detail (see figure 3.7), a backtracking step from level $k$ to level $k'$ is performed by exchanging the arcs of level $k'$ and $k-1$ and then continuing to investigate that level which is associated with arc $a_{k'}$. After changing the modulo parameter of $a_{k'}$, the current potential is still feasible for all chords $a_1,\ldots,a_{k'-1},a_{k'+1},\ldots,a_{k-1}$. Thus, a re-computation, as done by the PESP algorithm, of the values associated with the arcs $a_{k'+1},\ldots,a_{k-1}$ is not necessary any more.

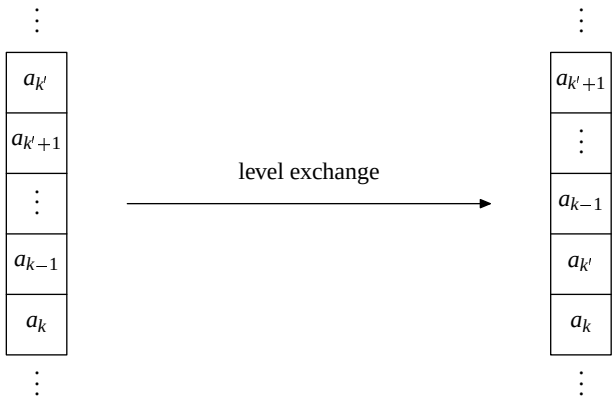



Figure 3.7: Exchange of arcs during a backtracking step

### Generalized Algorithm

The algorithm of Serafini and Ukovich can be generalized in some ways:

- *Choice of start tree $\mathcal{T}$*: Instead of choosing a minimum spanning tree concerning span lengths, an arbitrary start tree can be used. An example for a PESP instance where the Serafini and Ukovich start tree leads to a large search tree is given in figure 3.8.

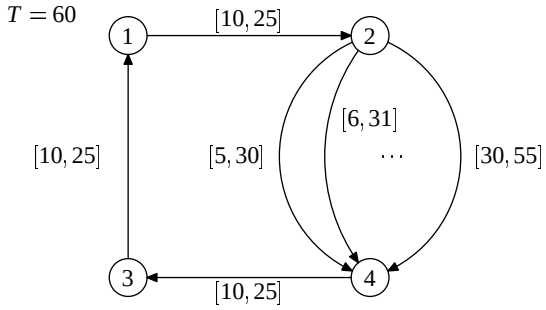



Figure 3.8: On this graph, the Serafini-Ukovich start tree should not be used.

If the start tree is chosen as the minimum spanning tree considering span lengths, none of the parallel arcs of this graph is chosen. As one can see from the parallel arcs, $\pi_4 - \pi_2 \equiv 30 \bmod T$ must be fulfilled. For some of these parallel arcs, several modulo parameters can be selected in such a way that a feasible potential can be found which cannot be extended to a feasible potential for the complete graph. In contrast to this, if one of the parallel arcs is chosen for the start tree, there is only one possibility for the modulo parameter of all parallel arcs and even for all arcs of the graph. No backtracking is needed.

In practice, the start tree suggested by Serafini and Ukovich provides comparably good results. The example of figure 3.8 seems to be very artificial. Moreover, our preprocessing methods eliminate all parallel arcs of the graph, as one can easily verify.

- *Choice of an arc to be examined at level k*: At level $k$, one can choose an arbitrary arc $a$ whose modulo parameter has not been fixed and look for a modulo parameter $z_a$. As we have already mentioned, Serafini and Ukovich suggest an examination order determined in advance. Schrijver and Steenbeek modify this order during the algorithm. We will discuss several arc choice rules in section 3.7 that lead to a considerable acceleration of the algorithm for our practical instances.

A generalized version of the algorithm of Serafini and Ukovich is given as algorithm 3.4.

## 3.7 Arc Choice for the Generalized Serafini-Ukovich Algorithm

We have already mentioned that the order by which the chords are chosen for modulo parameter fixing has much influence on the search tree and thus on the solution time of the algorithm of Serafini and Ukovich. In this section, we will give many new suggestions for this choice of chords, which often lead to better solution times or even to the solution of instances that could not be solved by the original algorithm because of lack of time.

**Algorithm 3.4** Generalized Serafini-Ukovich Algorithm

Choose a spanning tree $\mathcal{T}$. Set $S := A(\mathcal{T})$.
$z_a := 0$ for all $a \in S$
Determine a feasible potential $\pi$ for the graph $(V,S)$ with the fixed modulo parameters.
**loop**
  Choose a chord $a \in A \backslash S$.
  $Z_a := \{z_a \mid z_a$ is a feasible modulo parameter for $a$ if only arcs $a' \in S$ are considered, but with their fixed modulo parameters$\}$
  **if** $Z_a \neq \emptyset$ **then**
    Choose a $z \in Z_a$.
    $z_a := z$
    $S := S \cup \{a\}$
    Determine a feasible potential $\pi$ for the graph $(V,S)$ with the fixed modulo parameters.
    **if** $S = A$ **then**
      Stop. $\pi$ is a feasible potential for the PESP instance.
    **end if**
  **else**
    Perform backtracking: Choose another value from $Z_{a'}$ for the arc $a'$ whose modulo parameter was fixed in the iteration before. If this is not possible, perform further backtracking. Delete the corresponding arcs from $S$. If there are no more modulo parameters for the arc whose modulo parameter was fixed in the first iteration, stop. The PESP instance is infeasible.
  **end if**
**end loop**

#### Ordering by the Bounds for the Modulo Parameters in Advance

In order to heuristically keep the search tree "small", it has already been pointed out that the "most restrictive" chords should be selected as candidates for the fixing of modulo parameters first. This leads to the following idea: From proposition 3.3, we can derive an upper bound on the number of feasible modulo parameters for each chord. Instead of ordering the chords by increasing span length, one can order them by this upper bound. This leads to a remarkable acceleration of the algorithm in general (cf. the computational results in chapter 5). For example, when using this rule, the instance from figure 3.8 is solved without backtracking, even if the Serafini-Ukovich start tree is chosen.

A general disadvantage of a fixed ordering of chords in advance is that during the algorithm, only information concerning tree arcs and chords with fixed modulo parameters is used (except for the fact that the chords have been ordered). Even in the Schrijver/Steenbeek version, the ordering from the initialization has the main influence on the behavior of the algorithm.

#### Ordering by the Number of Modulo Parameters in each Search Tree Node

With the fixing of some modulo parameters, the bounds for the non-fixed modulo parameters may change. One can recalculate these bounds exactly after every fixing of a modulo parameter, i.e. in every node of the search tree, and choose the chord with the "most restrictive" bounds.

A naive implementation of this method would apply the standard Dijkstra algorithm twice to determine the bounds for the modulo parameter of a chord $a : i \to j$: In a first step, the algorithm starts with root node $i$. If the algorithm terminates with a label $\lambda(j)$, the tension $\pi_j - \pi_i$ can be raised at most by $\lambda(j)$ without changing already fixed modulo parameters (cf. appendix C). Starting with root node $j$, the resulting label $\lambda(j)$ is the amount by that the tension $\pi_j - \pi_i$ can be lowered. From these values, the possible modulo parameters $z_a$ can easily be found.

However, as soon as a chord $a$ with a modulo parameter *bound width* $\overline{z}_a - \underline{z}_a = 0$ or an infeasible chord $a$ is found, the bound determination in this search tree node can be stopped. By choosing chord $a$ for modulo parameter fixing (or backtracking), one can avoid creating additional branches in the search tree. Moreover, if the best bound width found so far is $w$, the $\text{Dij}^{\text{raise}}$ and $\text{Dij}^{\text{lower}}$ procedures, supplied with a proper value of $\delta$, can be used to decide whether the currently examined chord has a bound width of at least $w$ (the advantage of the $\text{Dij}^{\text{raise}}$ and $\text{Dij}^{\text{lower}}$ procedures is that in this case, the procedures probably terminate without having generated a complete shortest path tree).

By using this technique, often many modulo parameters can be fixed before another branching occurs in the search tree.

Our experiments have shown that it is often useful to examine the chords in a cyclic order: Let the chords with non-fixed modulo parameters be ordered as $a_1, \ldots, a_r$. If we stop the search at chord $a_\rho$ because there is only one feasible modulo parameter, then we can start the chord examination in the next iteration of algorithm 3.4 with chord $a_{\rho+1}$ and continue with chord $a_1$ after examining chord $a_r$. The complete algorithm for choosing a chord for modulo parameter fixing is given by algorithm 3.5.

#### Several Chords with the same Number of Feasible Modulo Parameters

If the minimum number of feasible modulo parameters $(w_* + 1)$ is $> 1$, it is useful to examine those chords with bound width $w_*$ more closely. In general, there are many chords with bound width $w_*$.

Assume that the chords $a_g$, $g \in \{1, \ldots, k\}$ have bound width $w_*$ (with $w_* > 0$). Let $z_g^0, \ldots, z_g^{w_*}$ be the feasible modulo parameters for chord $a_g$. Now, all subtrees corresponding to $z_g^h$, $g \in \{1, \ldots k\}$, $h \in \{0, \ldots, w_*\}$ are examined. If there are chords with modulo parameters leading to infeasibility before a branching occurs in the subtree, then choose a chord with the maximal number of such modulo parameters. Otherwise, let $d_{g,h}$ be the maximal number of nodes in the subtree corresponding to the modulo parameter $z_g^h$ before a further branching occurs (i.e. $w_* > 0$ again). Now let

$$d_g := \min_{w \in \{0,\ldots,w_*\}} d_{g,w} \qquad \text{and choose chord } a_{g_*} \text{ with } \qquad d_{g_*} = \max_{g \in \{1,\ldots,k\}} d_g.$$

This corresponds to "looking ahead" and then choosing the chord locally leading to a largest delay of further branchings an the subtree. $d_g$ is called *look-ahead value* of chord $a_g$. An example is shown in figure 3.9.

#### Maintaining a Candidate List for Look-Ahead

In general, there are too many chords with bound width $w_*$ for examining them all in a reasonable amount of time. Instead, a candidate list for chord investigation should be used.

**Algorithm 3.5** Choosing a Chord in the Generalized Serafini-Ukovich Algorithm

Let the chords with non-fixed modulo parameters be $a_1, \ldots, a_r$.
Let $a_s$ be the chord that was examined before a chord was chosen in the previous iteration of algorithm 3.4. If the first examined chord in the previous iteration was chosen, set $s := 0$.
$\rho := s+1$
$w_* := \infty$ /* $w$ is the current bound width, $w_*$ the best found bound width */
**while** $\rho \neq s$ **do**
  **if** $\rho > r$ **then**
    $\rho := 1$
  **end if**
  Let $a_\rho : i \to j$. Determine $z := \max\{z' \in \mathbb{Z} \mid \pi_j - \pi_i - z' \cdot T \geq l_\rho\}$.
  **if** $\pi_j - \pi_i - z \cdot T \leq u_\rho$ **then**
    $w := 0$
  **else**
    $w := -1$
  **end if**
  $\delta := l_\rho + (w_* - w) \cdot T - (\pi_j - \pi_i - z \cdot T)$
  Use $\text{Dij}^{\text{raise}}$ with parameter $\delta$ to determine the maximal amount $\lambda$ with $\lambda \leq \delta$ by which the tension $\pi_j - \pi_i$ can be raised without changing fixed modulo parameters.
  $w := w + \left\lfloor \dfrac{\lambda - l_\rho + \pi_j - \pi_i - z \cdot T}{T} \right\rfloor$
  **if** $w < w_*$ **then**
    Analogously, increase $w$ for lowering the tension.
    **if** $w < w_*$ **then**
      $\rho_* := \rho$; $w_* := w$
      **if** $w < 1$ **then**
        Stop. Choose chord $a_\rho$. /* $w + 1 = |Z_{a_\rho}| \leq 1$ */
      **end if**
    **end if**
  **end if**
  $\rho := \rho + 1$
**end while**
Stop. Choose chord $a_{\rho_*}$.

A simple strategy is to stop the look-ahead process after a limit of $k'$ chords, $k' < k$ and choose that chord $a_{g_*}$ with the best found value $d_{g_*}$ so far. Often, better results can be obtained for a dynamic look-ahead limit: The process is stopped after the product of the number $\kappa$ of already investigated arcs and the best found value $d_{g_*}$ exceeds a certain bound $D$.

Another heuristic approach is the following: For each chord $a_g$ that is to be examined, an *adjacency value* $c_g$ is determined. The value $c_g$ is defined as

$$c_g := \sum_{\substack{a \in S \\ a \text{ incident with } a_g}} T + l_a - u_a,$$

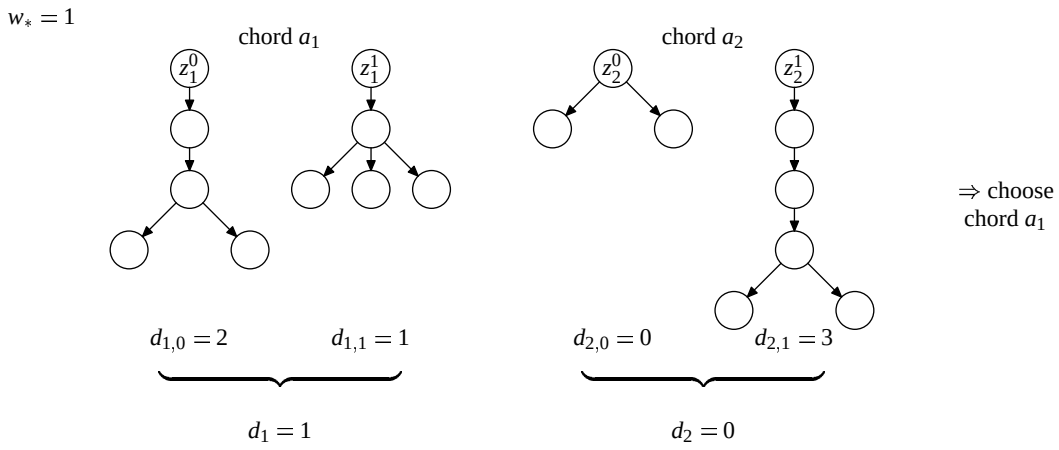



Figure 3.9: Determining a chord for modulo parameter fixing by "looking ahead"

where $S$ is the set of those arcs whose modulo parameters are already fixed (algorithm 3.4). $c_g$ is large if $a_g$ is incident with many arcs from $S$ and especially if those arcs have a small span. We can heuristically hope that a large value of $c_g$ corresponds to a "highly restrictive" chord $a_g$.

The adjacency value can be computed very fast, compared to look-ahead values. We can always choose the arc with highest adjacency value for modulo parameter fixing or even combine the two approaches: Starting with the chord with highest adjacency value, we only calculate look-ahead values for chords with relatively high adjacency value (say $\alpha \cdot c_g$, where $a_g$ is the chord with the highest look-ahead value so far, $\alpha < 1$). This strategy is given by algorithm 3.6.

**Algorithm 3.6** Choosing a Chord when $w_* > 0$

Let $\{a_1, \ldots, a_k\}$ be the set of chords with non-fixed modulo parameters. Let $c_1 \geq c_\kappa$ for all $\kappa \in \{2, \ldots, k\}$.
$q := 0$; $d_* := -1$; $c_* := c_1$; $g := 1$;
**loop**
  **if** $g > k$ **or** $q \cdot d_* \geq D$ **then**
    Stop. Choose chord $a_*$.
  **end if**
  **if** $|Z_{a_g}| = w_* + 1$ **then**
    $q := q + 1$
    **if** $c_g \geq \alpha \cdot c_*$ **then**
      **if** $d_g > d_*$ **or** ($d_g = d_*$ **and** $c_g > c_*$) **then**
        $d_* := d_g$; $c_* := c_g$; $a_* := a_g$
      **end if**
    **end if**
  **end if**
  $g := g + 1$
**end loop**

We have applied all of the methods from this section (and combinations) to a set of PESP instances from practice. From the results given in chapter 5, one can see that there are instances that could only be solved (or proven to be infeasible) with the help of these methods. On the other hand, the additional calculations take much time even for instances that can be solved with the original algorithm.

## 3.8 Polyhedral Structure of the PESP

In this section, we will examine the polyhedral structure of PESP. Recall the definitions from section 3.2:

$$Q = \{(\pi, z) \in \mathbb{R}^n \times \mathbb{Z}^m \mid l \le \mathfrak{N}^T \pi - Tz \le u\}$$
$$\mathcal{Z} = \{z \in \mathbb{Z}^m \mid \text{there is a } \pi \in \mathbb{R}^n \text{ such that } l \le \mathfrak{N}^T \pi - Tz \le u\}$$

Note that $\mathcal{Z}$ is the projection of $Q$ on the modulo parameters. The convex hull of $Q$ is called *unbounded timetable polyhedron*.

We already know that the modulo parameter can be fixed on the arcs of a fixed spanning tree $\mathcal{T}$ and thereby bounds $\underline{z}$ and $\overline{z}$ for the modulo parameters can be obtained (with $\underline{z}_a = \overline{z}_a = 0$ for tree arcs). As a consequence, we will also examine bounded versions of $Q$ and $\mathcal{Z}$:

$$Q_{\mathcal{T}}(\underline{z}, \overline{z}) := \{(\pi, z) \in \mathbb{R}^n \times \mathbb{Z}^m \mid l \le \mathfrak{N}^T \pi - Tz \le u,\ \underline{z} \le z \le \overline{z}\}$$
$$\mathcal{Z}_{\mathcal{T}}(\underline{z}, \overline{z}) := \{z \in \mathbb{Z}^m \mid \text{there is a } \pi \in \mathbb{R}^n \text{ such that } l \le \mathfrak{N}^T \pi - Tz \le u,\ \underline{z} \le z \le \overline{z}\}$$

$\operatorname{conv}(Q_{\mathcal{T}}(\underline{z}, \overline{z}))$ is called *bounded timetable polyhedron*.

In this section, we will derive new results for cutting planes of PESP instances and derive a new class of cutting planes for such instances.

### 3.8.1 The Unbounded Timetable Polyhedron

Let $\xi^T \pi + \varphi^T z \ge \varphi_0$ be a valid inequality for $\operatorname{conv} Q$. Without loss of generality assume that all spans $[l_a, u_a]$ fulfill $u_a - l_a < T$ (otherwise this span can be ignored). Let $(\pi, z) \in Q$. Let $k$ be a node and let $\mu \in \mathbb{Z}$. Then

$$\pi'_i := \begin{cases} \pi_i + \mu T & \text{if } i = k \\ \pi_i & \text{otherwise} \end{cases} \qquad \text{and} \qquad z'_a := \begin{cases} z_a + \mu & \text{if } a : i \to k \text{ for a node } i \\ z_a - \mu & \text{if } a : k \to i \text{ for a node } i \\ z_a & \text{otherwise} \end{cases}$$

lead to a feasible solution $(\pi', z') \in Q$. It is easy to see that $z' = z + \mu \theta_k$. Note that

$$\xi^T \pi' + \varphi^T z' = \xi^T \pi + \varphi^T z + \mu(\xi_k \cdot T + \varphi^T \theta_k). \tag{3.6}$$

**Proposition 3.5** *If $\xi^T \pi + \varphi^T z \ge \varphi_0$ is a valid inequality for $\operatorname{conv} Q$, then*

$$\xi_k \cdot T + \varphi^T \theta_k = 0 \qquad \text{for all nodes } k.$$

**Proof:** Let $(\pi, z) \in Q$. Assume that $\delta := \xi_k T + \varphi^T \theta_k < 0$ for a node $k$. Set $\rho := \xi^T \pi + \varphi^T z$ and $\mu := 1 + (\varphi_0 - \rho)/\delta$.

By the procedure above, $(\pi', z') \in Q$ can be generated. Now by (3.6)

$$\xi^T \pi' + \varphi^T z' = \xi^T \pi + \varphi^T z + \left(1 + \frac{\varphi_0 - \rho}{\delta}\right) \cdot \delta = \xi^T \pi + \varphi^T z + \delta + \varphi_0 - \rho = \varphi_0 + \delta < \varphi_0.$$

This contradicts $(\pi', z') \in Q$. An analogous contradiction can be found for $\delta > 0$. □

Define a projection

$$f : \begin{cases} \mathbb{R}^n \times \mathbb{R}^m & \to & \mathbb{R}^m \\ (\pi, z) & \mapsto & x := \sum_{k=1}^{n} (\pi_k \cdot \theta_k) - T \cdot z \end{cases}$$

and let $X := \{ f((\pi, z)) \mid (\pi, z) \in Q,\ 0 \le \pi < T \cdot \mathbf{1} \}$.

**Theorem 3.1** *$\xi^T \pi + \varphi^T z \ge \varphi_0$ is a valid inequality for* $\operatorname{conv} Q$ *if and only if $\xi_k = -\frac{1}{T} \varphi^T \theta_k$ for all nodes $k$ and $\varphi^T x \le -T \cdot \varphi_0$ is a valid inequality for* $\operatorname{conv} X$.

**Proof:** Let $\xi^T \pi + \varphi^T z \ge \varphi_0$ be a valid inequality for $\operatorname{conv} Q$. Then $\xi_k = -\frac{1}{T} \varphi^T \theta_k$ follows from proposition 3.5. The other condition is also fulfilled:

$$\begin{aligned} \xi^T \pi + \varphi^T z &\ge \varphi_0 \\ \sum_{k=1}^{n} -\varphi^T \frac{\theta_k}{T} \pi_k + \varphi^T z &\ge \varphi_0 \\ \varphi^T \left( \sum_{k=1}^{n} -\theta_k \pi_k + z \cdot T \right) &\ge T \cdot \varphi_0 \\ \varphi^T x &\le -T \cdot \varphi_0 \end{aligned}$$

Conversely, let $\xi_k = -\frac{1}{T} \varphi^T \theta_k$ for all nodes $k$ and let $\varphi^T x \le -T \cdot \varphi_0$ for $X$. Now consider an element $(\pi, z) \in Q$. We have to show $\xi^T \pi + \varphi^T z \ge \varphi_0$.

For each $(\pi, z) \in Q$, there exist uniquely determined integers $\mu_i$, $i \in \{1, \ldots, n\}$ such that $0 \le \pi_i + \mu_i \cdot T < T$. Setting $\pi'_i := \pi_i - \mu_i \cdot T$ and $z' := z - \sum_i \mu_i \theta_i$ leads to a feasible point $(\pi', x') \in Q$ and $x := f(\pi', z') \in X$. It follows that

$$\varphi_0 \le -\frac{1}{T} \varphi^T \left( \sum_{k=1}^{n} \pi'_k \theta_k - z' \cdot T \right) = \xi^T \pi' + \varphi^T z' =$$

$$\xi^T \pi + \varphi^T z - T \cdot \xi^T \mu - \varphi^T \cdot \sum_{k=1}^{n} \mu_k \theta_k = \xi^T \pi + \varphi^T z,$$

since $T \cdot \xi_k = -\varphi^T \theta_k$. The proof is complete. □

Regard $\varphi$ as a flow on the arcs of the graph $\mathcal{G}$. The amount $\varphi^T \theta_k$ can be interpreted as inflow minus outflow at node $k$, see figure 3.10. Now consider a valid inequality $\varphi^T z \ge \varphi_0$ for $\operatorname{conv} \mathcal{Z}$, which can be understood as induced by a valid inequality $\xi^T \pi + \varphi^T z \ge \varphi_0$ with $\xi = \mathbf{0}$. From theorem 3.1 it follows that $\varphi^T \theta_k = 0$ for all nodes $k$. This condition is known as *flow conservation law*. This gives the next theorem.

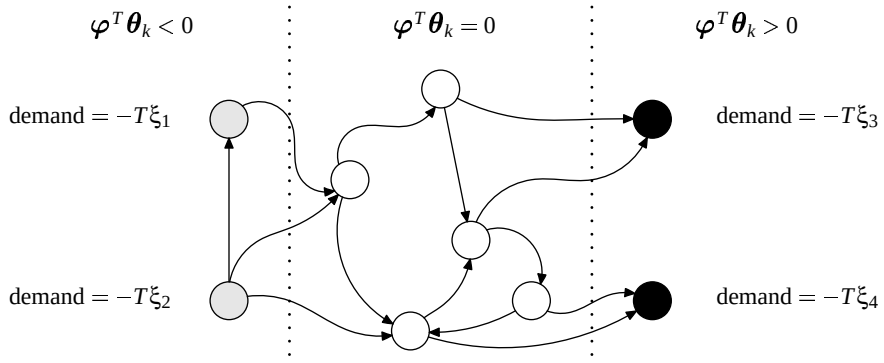



Figure 3.10: Flow conservation: $\xi_1+\xi_2+\xi_3+\xi_4 = 0$

**Theorem 3.2** *$\varphi^T z \geq \varphi_0$ is a valid inequality for* conv $\mathcal{Z}$*, if and only if $\varphi$ is a flow fulfilling $\mathfrak{N}^T\varphi = 0$ and*

$$\varphi_0 \leq \min\left\{\varphi^T z \mid z \in \mathcal{Z}\right\}.$$

### 3.8.2 Cycle Cutting Planes

An important class of cutting planes for PESP instances is given by the cycle cutting planes introduced as (3.3):

$$\gamma^T z \geq \left\lceil \frac{1}{T}\left(l^T\gamma^- - u^T\gamma^+\right)\right\rceil$$

for each elementary cycle with incidence vector $\gamma$.

### 3.8.3 Chain Cutting Planes

We will now introduce a new class of cutting planes. Consider a system of $m$ disjoint arcs between two nodes 1 and 2, where only lower bounds are given:

$$S := \left\{(\pi_1,\pi_2,z_1,\ldots,z_m) \in \mathbb{R}^2\times\mathbb{Z}^m \mid l_a \leq \pi_2-\pi_1-z_a\cdot T \text{ for all } a=1,\ldots,m\right\}$$

For each arc $a$ define

$$k_a := \frac{1}{T}((l_a) \bmod T - l_a) \in \mathbb{Z} \qquad \text{and} \qquad l'_a := (l_a) \bmod T.$$

Thus, $l'_a = k_a\cdot T + l_a$. Assume that $0 \leq l'_1 \leq l'_2 \leq \ldots l'_m < T$. For technical reasons, define $l'_0 := l'_m - T$. Set

$$\alpha_i := l'_i - l'_{i-1} \text{ for } i = 1,\ldots,m.$$

It is easy to see that the $\alpha$-values have the following properties:

(1) $0 \leq \alpha_j < T$ for each $j = 1,\ldots,m$

(2) $\sum_{j=1}^m \alpha_j = T$

(3) $\sum_{j=i+1}^m \alpha_j = l'_m - l'_i$

**Proposition 3.6** *For each $(\pi_1,\pi_2,z_1,\ldots,z_m)\in S$, the following inequality is valid:*

$$\pi_2-\pi_1 \geq l'_m + \sum_{j=1}^{m}\alpha_j(z_j-k_j) \tag{3.7}$$

**Proof:** Let $(\pi_1,\pi_2,z_1,\ldots,z_m)\in S$ with tension $x:=\pi_2-\pi_1$. Define $x':=(x) \bmod T$. Then $x'=x+kT$ for an integer $k$. Furthermore, there is a uniquely determined index $i\in\{0,\ldots,m\}$ fulfilling

$$0\leq l'_1\leq l'_2\leq\ldots\leq l'_i\leq x'<l'_{i+1}\leq\ldots\leq l'_m<T,$$

where $i=0$ means $x'<l'_1$.

From $l_j\leq x-z_jT$ we know that $l'_j=l_j+k_jT\leq x-z_jT+k_jT=x'+(k_j-k-z_j)T$. This implies $(k_j-k-z_j)\geq 0$ for all $j=1,\ldots,i$ and $(k_j-k-z_j)\geq 1$ for $j=i+1,\ldots,m$. Therefore,

$$\begin{aligned}
\pi_2-\pi_1 \;=\; x \;&=\; x+\sum_{j=1}^{m}\alpha_jz_j-\sum_{j=1}^{i}\alpha_jz_j-\sum_{j=i+1}^{m}\alpha_jz_j\\
&\geq\; x'-kT+\sum_{j=1}^{m}\alpha_jz_j+\sum_{j=1}^{i}\alpha_j(k-k_j)+\sum_{j=i+1}^{m}\alpha_j(k-k_j+1)\\
&=\; x'-kT+\sum_{j=1}^{m}\alpha_jz_j+k\sum_{j=1}^{m}\alpha_j-\sum_{j=1}^{m}\alpha_jk_j+\sum_{j=i+1}^{m}\alpha_j\\
&=\; x'+\sum_{j=1}^{m}\alpha_j(z_j-k_j)+l'_m-l'_i\\
&\geq\; l'_m+\sum_{j=1}^{m}\alpha_j(z_j-k_j)
\end{aligned}$$

This completes the proof. □

Now, we will discuss the case of an arbitrary graph $\mathcal{G}$ and a feasible point $(\boldsymbol{\pi},\boldsymbol{z})\in Q$. Consider two nodes 1 and 2 of $\mathcal{G}$ and $m$ paths from node 1 to node 2 with incidence vectors $\boldsymbol{p}_1,\ldots,\boldsymbol{p}_m$. For each $i\in\{1,\ldots,m\}$, define

$$\tilde{z}_i:=(\boldsymbol{p}_i^+)^T\boldsymbol{z}-(\boldsymbol{p}_i^-)^T\boldsymbol{z} \qquad\text{and}\qquad \tilde{l}_i:=(\boldsymbol{p}_i^+)^T\boldsymbol{l}-(\boldsymbol{p}_i^-)^T\boldsymbol{u}.$$

Obviously, we have $\tilde{l}_i\leq\pi_2-\pi_1-T\tilde{z}_i$ for each $i\in\{1,\ldots,m\}$. Using the box constraints (3.3) on the modulo parameters leads to $\tilde{z}_i\geq(\boldsymbol{p}_i^+)^T\underline{\boldsymbol{z}}-(\boldsymbol{p}_i^-)^T\overline{\boldsymbol{z}}=:\underline{\tilde{z}}_i$. From proposition 3.6, we obtain the valid inequality

$$\pi_2-\pi_1\geq\tilde{l}'_m+\sum_{j=1}^{m}\alpha_j(\underline{\tilde{z}}_j-k_j) \tag{3.8}$$

for $Q$. This inequality is called *chain cutting plane*.

For graph sizes given by practical instances, it is important to identify "small" path sets leading to "effective" chain cutting planes. This problem will be addressed in the following.

Suppose that we are given a system of $m$ paths from node 1 to node 2 and the chain cutting plane (3.8). The paths are denoted by $p_1,\ldots,p_m$ and correspond to incidence vectors $\boldsymbol{p}_1,\ldots,\boldsymbol{p}_m$. Now, exchange one path of this system, say $p_1$, by a path $q$ with incidence vector $\boldsymbol{q}$ and $\tilde{l}(\boldsymbol{q}):=(\boldsymbol{q}^+)^T\boldsymbol{l}-(\boldsymbol{q}^-)^T\boldsymbol{u}$ such

that $\tilde{l}_1 \equiv \tilde{l}(\boldsymbol{q}) \bmod T$. Together with the modified modulo parameter bound $\underline{\tilde{z}}'_1 := (\boldsymbol{q}^+)^T \underline{\boldsymbol{z}} - (\boldsymbol{q}^-)^T \boldsymbol{u}$, the modified integer $k'_1$ with $(\tilde{l}(\boldsymbol{q})) \bmod T = k'_1 T + \tilde{l}(\boldsymbol{q})$ yields the inequality

$$\pi_2 - \pi_1 \geq \tilde{l}'_m + \alpha_1(\underline{\tilde{z}}'_1 - k'_1) + \sum_{j=2}^{m} \alpha_j(\underline{\tilde{z}}_j - k_j).$$

Hence, the chain cutting plane gets tighter by the path exchange, if

$$\alpha_1(\underline{\tilde{z}}'_1 - k'_1) \geq \alpha_1(\underline{\tilde{z}}_1 - k_1) \qquad \text{or equivalently} \qquad \underline{\tilde{z}}'_1 - k'_1 \geq \underline{\tilde{z}}_1 - k_1.$$

Since $k_1 = \frac{1}{T}((\tilde{l}_1) \bmod T - \tilde{l}_1)$, $k'_1 = \frac{1}{T}((\tilde{l}(\boldsymbol{q})) \bmod T - \tilde{l}(\boldsymbol{q}))$ and $(\tilde{l}_1) \bmod T = (\tilde{l}(\boldsymbol{q})) \bmod T$, the path exchange improves the cutting plane, if and only if

$$\underline{\tilde{z}}'_1 + \frac{\tilde{l}(\boldsymbol{q})}{T} \geq \underline{\tilde{z}}_1 + \frac{\tilde{l}_1}{T},$$

which means $\underline{\tilde{z}}'_1 \cdot T + \tilde{l}(\boldsymbol{q}) \geq \underline{\tilde{z}}_1 \cdot T + \tilde{l}_1$. The best improvement of this exchange type can be found by solving a longest path problem: For a given value $\tau_i = (l_i) \bmod T$, we have to find a longest path $q$ from node 1 to node 2 with $(\tilde{l}(\boldsymbol{q})) \bmod T = \tau_i$, where the arc lengths are given by

$$\mu_a^+ := \underline{z}_a \cdot T + l_a \qquad \text{and} \qquad \mu_a^- := -\overline{z}_a \cdot T - u_a.$$

The *modulo path problem*

$$\begin{aligned} \max\{\mu(\boldsymbol{p}) \mid\ & \boldsymbol{p} \text{ incidence vector for a path from node 1 to node 2 and} \\ & ((\boldsymbol{p}^+)^T \boldsymbol{l} - (\boldsymbol{p}^-)^T \boldsymbol{u}) \bmod T = \tau\} \end{aligned} \tag{3.9}$$

can, for fixed period $T$, be solved in polynomial time by the following dynamic programming formulation:

For each node $i$ and each modulo value $\tau = 0, \ldots, T-1$, define

$$\begin{aligned} F_k(\tau, i) := \max\{\mu(\boldsymbol{p}) \mid\ & \boldsymbol{p} \text{ incidence vector of a path from 1 to } i \text{ with } k \text{ arcs and} \\ & ((\boldsymbol{p}^+)^T \boldsymbol{l} - (\boldsymbol{p}^-)^T \boldsymbol{u}) \bmod T = \tau\} \end{aligned}$$

Starting with

$$F_0(\tau, i) := \begin{cases} 0 & \text{if } \tau = 0 \text{ and } i = 1 \\ \infty & \text{otherwise,} \end{cases}$$

we obtain

$$\begin{aligned} F_{k+1}(\tau, i) = \max\{ & \max \{F_k(\tau', j) + \mu_a^+ \mid a : j \to i \text{ and } l_a + \tau' \equiv \tau \bmod T\}, \\ & \max \{F_k(\tau', j) + \mu_a^- \mid a : i \to j \text{ and } -u_a + \tau' \equiv \tau \bmod T\}\}. \end{aligned}$$

### 3.8.4 Simple Lifting Procedures for Flow Inequalities

The flow inequalities $\varphi^T \boldsymbol{z} \geq \varphi_0$ may contain coefficients $\varphi_i \notin \{-1, 0, 1\}$. In many of those cases, the defined bounds on the modulo parameters allow a simple *coefficient reduction* (or *lifting*) procedure. A short view on coefficient reduction and its effect on the solution of MIPs is given in appendix B.

Without loss of generality assume that $0 \leq z_a \leq \overline{z}_a$ and $\varphi_a \geq 0$ (otherwise, use the transformation $z'_a := z_a - \underline{z}_a$ for arcs with $\underline{z}_a \neq 0$ and $\varphi_a \geq 0$, and the transformation $z'_a := \overline{z}_a - z_a$ for arcs with $\underline{z}_a \neq 0$ and $\varphi_a < 0$; this leads to an inequality of the desired type, which can be transformed back after the lifting).

Let $\varphi_1, \ldots, \varphi_k > 0$ and $\varphi_i = 0$ for $i > k$. Moreover, assume $\rho := \gcd\{\varphi_1, \ldots, \varphi_k\} = 1$. Otherwise, the inequality may be tightened by

$$\frac{1}{\rho} \varphi^T \boldsymbol{z} \geq \left\lceil \frac{\varphi_0}{\rho} \right\rceil .$$

For $k = 2$ and $\varphi_0 > 0$, the inequality $\varphi_1 z_1 + \varphi_2 z_2 \geq \varphi_0$ can be lifted to

$$\left\lceil \frac{\varphi_0}{\varphi_2} \right\rceil z_1 + \left\lceil \frac{\varphi_0}{\varphi_1} \right\rceil z_2 \geq \left\lceil \frac{\varphi_0^2}{\varphi_1 \cdot \varphi_2} \right\rceil .$$

As an example, the inequality $z_1 + 2z_2 \geq 1$ can be transformed to $z_1 + z_2 \geq 1$ by this method. A fractional solution like $(0, \frac{1}{2})$ is infeasible for the transformed inequality.

The lifting procedure can be applied successively to each variable in the following way: Define $\varphi'_2 := \gcd\{\varphi_2, \ldots, \varphi_k\}$ and

$$z'_2 := \frac{1}{\varphi'_2} \sum_{a=2}^{k} \varphi_a z_a .$$

This yields the inequality $\varphi_1 z_1 + \varphi'_2 z'_2 \geq \varphi_0$. Now, one can apply the lifting procedure to this equation and continue by selecting all other variables $z_2, \ldots, z_k$.

### 3.8.5 Single Bound Improvement

For many combinatorial PESP algorithms, it is easy to use additional information on the bounds of the modulo parameters, while it may be difficult to use information from general cutting planes. Therefore, we will now focus on the generation of modulo parameter bounds by cutting planes for $\mathcal{Z}$.

Assume that $\beta^T \boldsymbol{z} \geq \beta_0$ is a cutting plane for $\mathcal{Z}$. Consider an arc $a$ with $\beta_a > 0$ and define $\beta_a := \beta - \beta_a \cdot \boldsymbol{e}^a$. Then

$$\beta_a z_a \geq \beta_0 - (\beta_a^+)^T \overline{\boldsymbol{z}} + (\beta_a^-)^T \underline{\boldsymbol{z}}.$$

In case of

$$\left\lceil \frac{1}{\beta_a} \left( \beta_0 - (\beta_a^+)^T \overline{\boldsymbol{z}} + (\beta_a^-)^T \underline{\boldsymbol{z}} \right) \right\rceil > \underline{z}_a \tag{3.10}$$

we obtain an improved lower bound for the modulo parameter of arc $a$.

A *single bound separation algorithm* for a class $\mathcal{C}$ of cutting planes is a method to find, for an arc $a$, either a cutting plane $(\beta^T \boldsymbol{z} \geq \beta_0) \in \mathcal{C}$ which improves the current bounds for the modulo parameter $z_a$ according to (3.10) or to prove that the bounds for $z_a$ cannot be improved by a cutting plane from $\mathcal{C}$.

Define $\mathcal{C}^{\mathrm{b}}$ as the class of all single bound inequalities resulting from (3.10). Adding all inequalities from $\mathcal{C}^{\mathrm{b}}$ to the polytope $\mathrm{conv}\,(Q_{\mathcal{I}}(\underline{z},\overline{z}))$ (possibly) yields strengthened bounds $\underline{z}' \geq \underline{z}$ and $\overline{z}' \leq \overline{z}$ for the modulo parameters. This gives a tighter relaxation polytope

$$\mathrm{conv}\,(Q_{\mathcal{I}}(\underline{z}',\overline{z}')) =: \mathcal{C}^{\mathrm{b}}(\mathrm{conv}\,(Q_{\mathcal{I}}(\underline{z},\overline{z}))).$$

If $\mathrm{conv}\,(Q_{\mathcal{I}}(\underline{z}',\overline{z}')) \neq \emptyset$, we can again apply the single bound cutting planes to the improved bounds. Proceeding in a recursive manner, we finally obtain either an empty polytope or a polytope where no bound can be improved. This polytope is called the $\mathcal{C}^{\mathrm{b}}$-*kernel.* An example for the calculation of such a $\mathcal{C}^{\mathrm{b}}$-kernel is shown in figure 3.11.

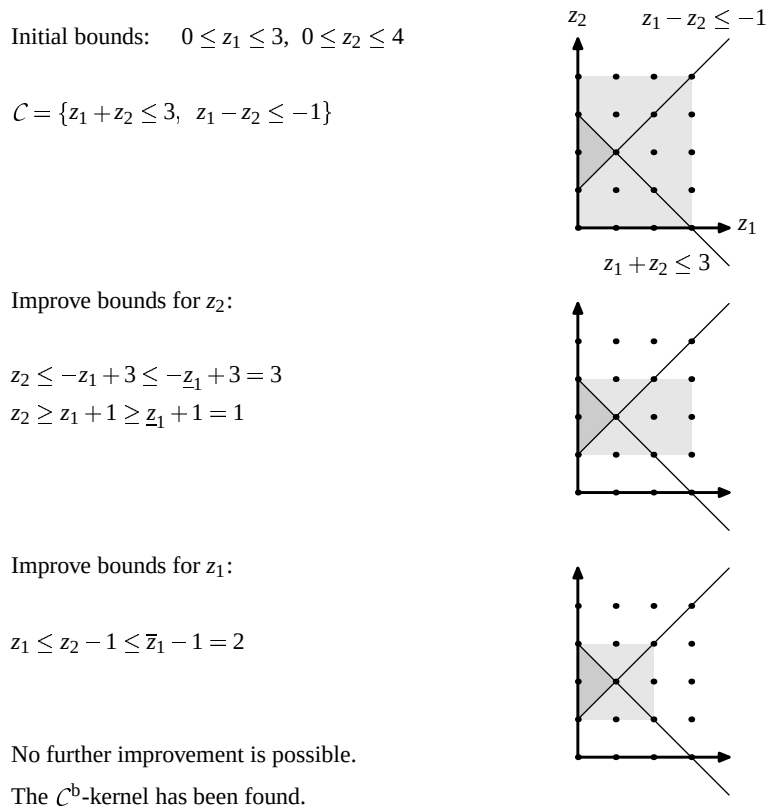



Figure 3.11: Calculation of the $\mathcal{C}^{\mathrm{b}}$-kernel

**Theorem 3.3** *Let the time period $T$ be fixed. If the single bound separation problem for a class $\mathcal{C}$ of cutting planes is polynomially soluble, then the $\mathcal{C}^{\mathrm{b}}$-kernel can be calculated in polynomial time.*

**Proof:** The number of possible modulo parameters $\overline{z}_a - \underline{z}_a + 1$ for a chord $a$ is bounded by $n+1$. This follows from the fact that each chord generates a uniquely determined cycle in the spanning tree. This cycle has at most $n$ arcs. Since we have $u_a - l_a < T$ for each span $[l_a, u_a]$, the bound difference $\overline{z}_a - \underline{z}_a$ cannot be larger than $n$.

During the kernel calculation we only can improve $m \cdot n$ bounds (otherwise we get an empty polytope). If the bound separation problem can be solved in polynomial time with complexity $f(m,n)$, the calculation of the kernel can be done within complexity $m \cdot n \cdot f(m,n)$. □

**Proposition 3.7** *The single bound separation problem for the class of cycle inequalities for an arc $a$ : $i \to j$ can be solved by calculating a shortest path from node $j$ to node $i$ with the following objective ($\boldsymbol{p}$ is the incidence vector of the path):*

$$(\boldsymbol{p}^+)^T(\boldsymbol{u}+T\overline{\boldsymbol{z}}) - (\boldsymbol{p}^-)^T(\boldsymbol{l}+T\underline{\boldsymbol{z}}).$$

**Proof:** Consider a cycle inequality $\boldsymbol{\gamma}^T\boldsymbol{z} \geq \gamma_0$. Let $a : i \to j$ be an arc of the corresponding cycle, i.e. $\gamma_a \neq 0$. Then $\boldsymbol{\gamma} =: \boldsymbol{p} + \boldsymbol{e}^a$, where $\boldsymbol{p}$ is the incidence vector of a path from node $j$ to node $i$. From (3.2), we have

$$\begin{aligned} 0 &\leq (\boldsymbol{\gamma}^+)^T(\boldsymbol{u}+T\boldsymbol{z}) - (\boldsymbol{\gamma}^-)^T(\boldsymbol{l}+T\boldsymbol{z}) \\ &= Tz_a + u_a + (\boldsymbol{p}^+)^T(\boldsymbol{u}+T\boldsymbol{z}) - (\boldsymbol{p}^-)^T(\boldsymbol{l}+T\boldsymbol{z}) \\ &\leq Tz_a + u_a + (\boldsymbol{p}^+)^T(\boldsymbol{u}+T\overline{\boldsymbol{z}}) - (\boldsymbol{p}^-)^T(\boldsymbol{l}+T\underline{\boldsymbol{z}}) \end{aligned}$$

This yields the inequality

$$z_a \geq \left\lceil \frac{1}{T}\left(-u_a - (\boldsymbol{p}^+)^T(\boldsymbol{u}+T\overline{\boldsymbol{z}}) + (\boldsymbol{p}^-)^T(\boldsymbol{l}+T\underline{\boldsymbol{z}})\right)\right\rceil.$$

The right hand side of this inequality only depends on the length of the path belonging to $\boldsymbol{p}$ with respect to the arc lengths from the proposition. In order to get a tight bound for $z_a$, the right hand side has to be maximized, which corresponds to finding a shortest path according to the arc weights from the proposition. □

### 3.8.6 Flow Inequalities and Single Bound Improvement

We will now examine single bound improvement by cutting planes $\boldsymbol{\varphi}^T\boldsymbol{z} \geq \varphi_0$ where $\boldsymbol{\varphi}$ is a flow with $\mathfrak{N}^T\boldsymbol{\varphi} = 0$ (cf. section 3.8.1). Such an inequality is called *flow inequality.* Consider the example of figure 3.12.

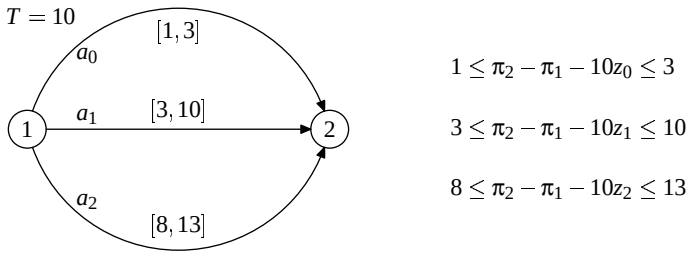



Figure 3.12: Example instance

The cycle cutting planes for the example lead to

$$\begin{array}{rcccl} -9 & \leq & 10(z_1-z_0) & \leq & 0 \\ -12 & \leq & 10(z_2-z_0) & \leq & -5 \\ -2 & \leq & 10(z_1-z_2) & \leq & 10 \end{array} \quad \text{or equivalently} \quad \begin{array}{rcccl} & & z_1-z_0 & = & 0 \quad (*) \\ & & z_2-z_0 & = & -1 \quad (**) \\ 0 & \leq & z_1-z_2 & \leq & 1. \end{array}$$

The resulting lower bound inequalities for the variable $z_1$ are:

$$z_1 \geq \underline{z}_0 \qquad \text{and} \qquad z_1 \geq \underline{z}_2$$

Combining $(*)$ and $(**)$ leads to $z_1 - z_2 = 1$, which gives the better bound

$$z_1 \geq \underline{z}_2 + 1,$$

which cannot be generated by the single bound separation algorithm for cycle inequalities. This bound can only be found by combining cycles. This idea will be generalized in the following.

Consider a set $I$ of cycles and the corresponding incidence vectors $\gamma_i$, $i \in I$. Let the resulting cycle inequalities be denoted by $\underline{\alpha}_i \leq \gamma_i^T \boldsymbol{z} \leq \overline{\alpha}_i$. Let $a = 1$ be a fixed arc, and $\boldsymbol{\varphi}^T \boldsymbol{z} \geq \varphi_0$ be a flow inequality with $\varphi_1 > 0$.

Define

$$\varphi_a^+ := \begin{cases} \varphi_a & \text{if } \varphi_a \geq 0 \\ 0 & \text{otherwise} \end{cases} \qquad \text{and} \qquad \varphi_a^- := \begin{cases} -\varphi_a & \text{if } \varphi_a < 0 \\ 0 & \text{otherwise.} \end{cases}$$

The induced single bound flow inequality for variable $z_1$ is given by

$$z_1 \geq \left\lceil \frac{1}{\varphi_1} \left( \varphi_0 - \sum_{a \neq 1} \varphi_a^+ \overline{z}_a + \sum_{a \neq 1} \varphi_a^- \underline{z}_a \right) \right\rceil.$$

For simplicity we assume bounds $\underline{z}_a := 0 \leq z_a \leq \overline{z}_a$ for all arcs (a corresponding transformation was given in section 3.8.4). Now, assume that the flow $\boldsymbol{\varphi}$ is generated by the above introduced system of elementary cycles, where on each cycle with incidence vector $\gamma_i$, the amount $\varepsilon_i$ is circulating. Then

$$\boldsymbol{\varphi} = \sum_{i \in I} \varepsilon_i \gamma_i,$$

and by combining the cycle inequalities we obtain

$$\boldsymbol{\varphi}^T \boldsymbol{z} \geq \varphi_0 := (\varepsilon^+)^T \underline{\boldsymbol{\alpha}} - (\varepsilon^-)^T \overline{\boldsymbol{\alpha}}.$$

Without loss of generality assume that $\varphi_1 = 1$. The resulting single bound inequality is then given by

$$z_1 \geq (\varepsilon^+)^T \underline{\boldsymbol{\alpha}} - (\varepsilon^-)^T \overline{\boldsymbol{\alpha}} - \sum_{a \neq 1} \varphi_a^+ \overline{z}_a + \sum_{a \neq 1} \varphi_a^- \underline{z}_a.$$

## 3.9 Branch-and-Cut Method

In this section we describe a PESP algorithm which uses the ideas of the Serafini-Ukovich algorithm and combines it with the polyhedral results of section 3.8. The basic idea is to use a branch-and-cut (see appendix B) method applied to the timetable polyhedron $\operatorname{conv}(Q_{\mathcal{I}})$.

The method starts with $\underline{\boldsymbol{z}}$ and $\overline{\boldsymbol{z}}$ from proposition 3.3. Then, the following principle is applied: Consider a relaxation set $\tilde{Q}_{\mathcal{I}}(\underline{\boldsymbol{z}}, \overline{\boldsymbol{z}}) \supseteq Q_{\mathcal{I}}(\underline{\boldsymbol{z}}, \overline{\boldsymbol{z}})$ for which the decision problem $\tilde{Q}_{\mathcal{I}}(\underline{\boldsymbol{z}}, \overline{\boldsymbol{z}}) \stackrel{?}{=} \emptyset$ should be easy. In case of $\tilde{Q}_{\mathcal{I}}(\underline{\boldsymbol{z}}, \overline{\boldsymbol{z}}) \neq \emptyset$, we obtain an element $(\boldsymbol{\pi}, \tilde{\boldsymbol{z}}) \in \tilde{Q}_{\mathcal{I}}(\underline{\boldsymbol{z}}, \overline{\boldsymbol{z}})$. If $\tilde{\boldsymbol{z}}$ is integral, we have found a solution

for the PESP instance. Otherwise, we pick a fractional modulo parameter $\tilde{z}_a \notin \mathbb{Z}$ and generate two problems with disjunctive solution spaces by demanding $z_a \leq \lfloor \tilde{z}_a \rfloor$ or $z_a \geq \lceil \tilde{z}_a \rceil$.

The subproblems only differ from the original problem by new bounds. A binary search tree like in figure 3.13 is obtained. At each node of the search tree, single bound improvement cuts can be applied.

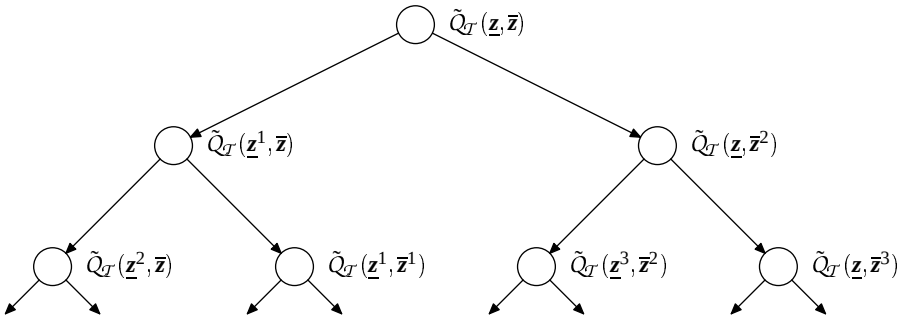



Figure 3.13: Binary search tree for the branch-and-cut method

The performance of the algorithm (i.e. the size / shape of the search tree) depends on many points, including these:

- In general, there is more than one fractional modulo parameter. We have to decide on which of those to branch.
- If $\tilde{Q}_T(\underline{z},\overline{z}) \neq \emptyset$, the determination of a point $(\pi,z) \in \tilde{Q}_T(\underline{z},\overline{z})$ can be done in different ways. Another idea is to add heuristics for minimizing the number of fractional modulo parameters to this determination algorithm.

For our branch-and-cut method, we will use the convex hull of the following enlarged feasible set:

$$\tilde{Q}_T(\underline{z},\overline{z}) := \{(\pi,z) \in \mathbb{R}^n \times \mathbb{R}^m \mid l \leq \mathfrak{N}^T\pi - Tz \leq u,\ \underline{z} \leq z \leq \overline{z}\}$$

**Proposition 3.8** *For $0 \leq u_a - l_a < T$ and integral bounds $\underline{z}_a \leq \overline{z}_a$ consider the decision problem*

$$\mathrm{conv}\,(\tilde{Q}_T(\underline{z},\overline{z})) \stackrel{?}{=} \emptyset. \tag{3.11}$$

*If $\underline{z} \not\leq \overline{z}$, the set is empty. Otherwise, (3.11) is equivalent to the feasible differential problem*

$$\Pi(\underline{z},\overline{z}) := \{\pi \mid l + T\underline{z} \leq \mathfrak{N}^T\pi \leq u + T\overline{z}\} \stackrel{?}{=} \emptyset.$$

**Proof:** We will show that

$$\left\{ \pi \;\middle|\; \bigvee_z (\pi,z) \in \mathrm{conv}\,(\tilde{Q}_T(\underline{z},\overline{z})) \right\} = \Pi(\underline{z},\overline{z}).$$

The implication $(\pi,z) \in \mathrm{conv}\,(\tilde{Q}_T(\underline{z},\overline{z})) \Rightarrow l + T\underline{z} \leq \mathfrak{N}^T\pi \leq u + T\overline{z}$ is obvious.

Conversely, let $\pi \in \mathbb{R}^n$ with $l + T\underline{z} \le \Pi^T \pi \le u + T\overline{z}$. Define

$$\alpha := -\frac{1}{T}(u - \Pi^T \pi) \qquad \text{and} \qquad \beta := -\frac{1}{T}(l - \Pi^T \pi).$$

Now $\alpha \le \overline{z}$, $\beta \ge \underline{z}$ and $\alpha \le \beta$. Together with $\underline{z} \le \overline{z}$ this shows

$$\max\{\alpha, \underline{z}\} \le \min\{\beta, \overline{z}\}.$$

For each $\tilde{z}$ with $\max\{\alpha, \underline{z}\} \le \tilde{z} \le \min\{\beta, \overline{z}\}$, we obtain $\underline{z} \le \tilde{z} \le \overline{z}$ and $\alpha \le \tilde{z} \le \beta$, or equivalently $l \le \Pi^T \pi - T\tilde{z} \le u$. Hence, $(\pi, \tilde{z}) \in \operatorname{conv}(\tilde{Q}_{\mathcal{I}}(\underline{z}, \overline{z}))$. □

From this proof it follows that $(\pi, z) \in \operatorname{conv}(\tilde{Q}_{\mathcal{I}}(\underline{z}, \overline{z}))$ if and only if $\pi \in \Pi(\underline{z}, \overline{z})$ and

$$\tilde{z}_a^l(\pi) := -\frac{1}{T}(u_a + T\overline{z}_a - (\pi_j - \pi_i)) \le z_a \le -\frac{1}{T}(l_a + T\underline{z}_a - (\pi_j - \pi_i)) =: \tilde{z}_a^u(\pi).$$

This means that a potential $\pi \in \Pi(\underline{z}, \overline{z})$ is feasible with arc $a : i \to j$, if the interval $[\tilde{z}_a^l, \tilde{z}_a^u]$ contains at least one integer value $z_a \in \mathbb{Z}$. Otherwise $\pi$ is infeasible for this arc. We will use this fact to apply the heuristic method of algorithm 3.7 to minimize the number of fractional modulo parameters.

The complete branch-and-cut method is described as algorithm 3.8. As we have already mentioned, pure linear (mixed integer) programming approaches seem to be inadequate for the solution of PESP instances. For this reason we use the feasible differential problem relaxation. In order to maintain this problem structure at every node of the search tree, we use only cutting planes compatible with the structure. The single bound cuts from section 3.8.5 have this property.

For a practical implementation of the branch-and-cut method, the data structures used for representing the tree have to be chosen carefully. There is no obvious way for a sufficient list structure like in [59].

Some concepts from section 3.7 can be adapted to the branch-and-cut method. As an example, a look-ahead value for each arc with a fractional modulo parameter is given by the amount of bound reduction during the subsequent $\mathcal{C}^{\mathrm{b}}$-kernel calculation. Since this calculation is a time consuming process, it is important to reduce the number of arcs with fractional modulo parameters for this approach in particular.

**Algorithm 3.7** Minimizing Fractional Values

Let $\pi$ be a (not necessarily feasible) potential.
$F := \{a \in A \mid \pi \text{ is feasible with } a\}$
$R := A \backslash F$
**for** each $a \in F$ **do**
Determine the unique integer $z_a$ with

$$l_a + T\underline{z}_a \leq l_a + Tz_a \leq \pi_j - \pi_i \leq u_a + Tz_a \leq u_a + T\overline{z}_a.$$

**end for**
**for** each $a \in A$ **do**

$$\underline{d}_a := \begin{cases} l_a + Tz_a & \text{if } a \in F \\ l_a + T\underline{z}_a & \text{otherwise} \end{cases} \qquad \overline{d}_a := \begin{cases} u_a + Tz_a & \text{if } a \in F \\ u_a + T\overline{z}_a & \text{otherwise} \end{cases}$$

**end for**
**while** $R \neq \emptyset$ **do**
Choose $a \in R$.
$R := R \backslash \{a\}$
Let $\delta_l, \delta_u$ be the amount for which the tension has to be lowered or raised to make the potential feasible.
**if** $\text{Dij}^{\text{lower}}(\delta_l, \underline{\boldsymbol{d}}, \overline{\boldsymbol{d}}, \pi, \gamma)$ or $\text{Dij}^{\text{raise}}(\delta_r, \underline{\boldsymbol{d}}, \overline{\boldsymbol{d}}, \pi, \gamma)$ succeeds **then**
Update the potential according to the solution.
Update the bounds for $a$ by $\underline{d}_a := l_a + Tz_a$ and $\overline{d}_a := u_a + Tz_a$.
**end if**
**end while**
Stop.

**Algorithm 3.8** Branch-and-Cut Method with FDP Relaxation

$\mathcal{L} := \{Q_{\mathcal{I}}(\underline{\mathbf{z}}, \overline{\mathbf{z}})\}$
**loop**
  **if** $\mathcal{L} = \emptyset$ **then**
    Stop. The instance is infeasible.
  **end if**
  Choose $Q \in \mathcal{L}$. Let the modulo parameter bounds of $Q$ be denoted by $\underline{\mathbf{z}}, \overline{\mathbf{z}}$.
  $\mathcal{L} := \mathcal{L} \setminus \{Q\}$
  Add single bound improving cuts to $Q$. /* calculate the $\mathcal{C}^{\mathrm{b}}$-kernel */
  **if** $\mathrm{conv}(\tilde{Q}) = \emptyset$ **then**
    **continue** /* FDP relaxation infeasible */
  **end if**
  Choose $(\boldsymbol{\pi}', \mathbf{z}') \in \mathrm{conv}(\tilde{Q})$.
  Try to reduce the number of fractional values of $\mathbf{z}'$. Let the updated solution be given by $(\boldsymbol{\pi}, \mathbf{z})$.
  **if** $\mathbf{z} \in \mathbb{Z}^m$ **then**
    Stop. $(\boldsymbol{\pi}, \mathbf{z})$ is a feasible solution for the instance.
  **end if**
  Choose an arc $a \in A$ with $z_a \notin \mathbb{Z}$ to branch on.
  Generate two new problems $Q^1$ and $Q^2$ from $Q$ with adjusted bounds

$$\overline{z}_a^1 := \lfloor z_a \rfloor \qquad \text{and} \qquad \underline{z}_a^2 := \lceil z_a \rceil.$$

  $\mathcal{L} := \mathcal{L} \cup \{Q^1, Q^2\}$
**end loop**

# Chapter 4

# Cost Optimal Schedules

In this chapter, solution methods for the MIP formulation of the minimum cost scheduling problem from section 2.8, MIP-MCSP, are presented. Since a direct MIP solution with a commercial solver turns out to be impossible for practical problem instances, a decomposition approach is developed.

The decomposition is integrated into a relaxation iteration algorithm and into a branch-and-bound algorithm. Fast methods for solving the subproblems arising from the decomposition are given. At the end of this chapter, the algorithms are extended in such a way that a certain nonlinear version of the model for minimum cost scheduling can be handled.

## 4.1 Mixed Integer Programming

A straightforward attempt to solve the MIPs for minimum cost train scheduling of section 2.8 (cf. figure 2.10) is the direct use of commercial MIP solvers on the problem instances. For practical instances (such as the InterCity network of Germany), the solution of the MIPs may take several days. In many cases, it is not even possible to find optimal solutions with our computer hardware and software (cf. section 5.2).

As we have already mentioned, our intention is to develop a model that can be used for strategic planning. In order to analyze or compare different scenarios, the solution times should not exceed a few minutes for practical instances. In the following sections, we will develop strategies that will enable us to solve instances or at least to find solutions of practical interest within such a time horizon.

## 4.2 Problem Decomposition

Two widely used classes of solution methods for solving MIPs (see appendix B) are:

- branch-and-bound methods
- iterative relaxation methods (e.g. cutting plane methods)

Both classes require solving a relaxation of the respective MIP. The typically applied relaxation is the LP relaxation (cf. appendix B). This is in particular the case for the commercial MIP solver CPLEX, which we have used.

For the minimum cost scheduling problem, we will develop another type of relaxation leading to much better results concerning solution time.

Consider the coefficient matrix of the objective function and the constraints for the MIP-MCSP, figure 4.1. The gray color indicates that there are nonzero coefficients for the respective variable and the corresponding class of constraints.

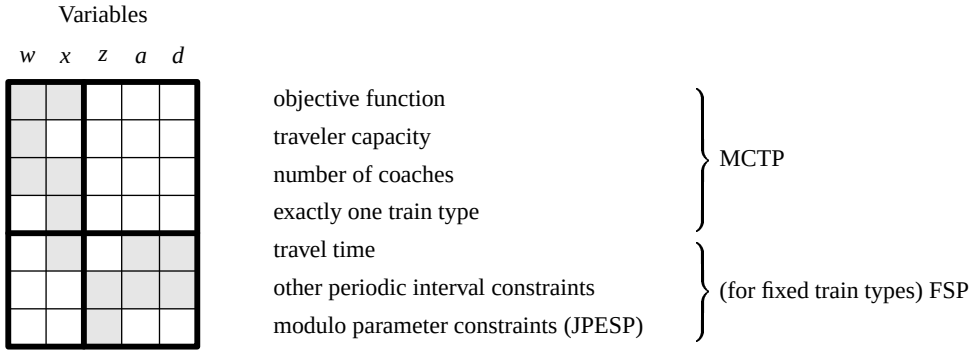



Figure 4.1: Structure of objective function and constraints of the MCSP

With the exception of the $x$-variables in the travel time constraints, the matrix can be divided into a two-block diagonal matrix. One block represents the problem of minimizing the cost considering only the constraints for capacity, number of coaches and selection of one train type. This problem was called minimum cost type problem (MCTP) in section 2.9. The integer programming formulation of this problem, which is given by this block, will be called IP-MCTP in the following. The problem of satisfying the constraints of the other block *for fixed train types* will be called *feasible schedule problem (FSP)*. The FSP is a JPESP (cf. section 2.5).

If the train types are fixed, the MCTP and the FSP can be solved separately. The solution of the FSP does not influence the objective value, because the corresponding variables are not in the objective function. If the FSP is feasible, the optimal solution of the MCTP and the feasible solution for the FSP can be combined to an optimal solution for the complete problem.

We will use the MCTP as a relaxation for the MCSP. Based on this relaxation, we propose a relaxation iteration algorithm in section 4.3 and a branch-and-bound algorithm in section 4.4. In both cases, we will have to solve many instances of the MCTP and the FSP. Therefore, fast solution techniques for these problems are developed in section 4.5 (for the MCTP) and section 4.6 (for the FSP).

## 4.3 Relaxation Iteration Method

For a short introduction on mixed integer programs and solution methods we refer to appendix B. We will now develop a relaxation iteration method for solving MCSP instances. From appendix B we know several crucial points for the design of such an algorithm:

- *Choice of the relaxation*: As initial relaxation for the iteration method for a MIP-MCSP instance we use the corresponding MCTP instance. This can, for example, be interpreted as an integer program of considerably smaller size (IP-MCTP). By using the ideas of section 4.5, we will be able to solve the MCTP instance for practical networks with a commercial MIP solver in a few seconds.

  The main disadvantage of the IP-MCTP relaxation is the fact that after reducing the solution space in an iteration, we will have to restart the MIP solution process completely (there is no such obvious idea like the dual simplex algorithm for LP relaxation iteration). Therefore, it will be important to keep the number of iterations small.

- *Feasibility check*: Let the optimal solution of the relaxation be given by the vector $\boldsymbol{x}$ of the train type variables and the vector $\boldsymbol{w}$ of variables for numbers of coaches. We now need to find out whether the FSP constraints can be satisfied with $\boldsymbol{x}$. This problem is a JPESP. A simple approach to solve JPESP instances is given by mixed integer programming (with an arbitrary objective function — only feasibility is important). In section 4.6, we will develop an algorithm for JPESP instances based on the PESP algorithm by Serafini and Ukovich (cf. chapter 3), which will have certain advantages.

- *Reduction of the solution space*: If the FSP instance of some iteration is infeasible, at least one of the train types has to be changed in order to get a feasible solution of the MCSP instance. Let $\tau_r$ be the train type for line $r \in \mathcal{R}$ in the optimal solution of the relaxation. Then, the following linear constraint for the train type variables is introduced:

$$\sum_{r \in \mathcal{R}} x_{r,\tau_r} \leq |\mathcal{R}| - 1 \tag{4.1}$$

  This inequality can be added to the IP-MCTP instance in the subsequent iteration.

Since there are many possible combinations of train types in practical instances, we may have to add a lot of inequalities to the original MIP-MCTP instance, eventually slowing down the MIP solution process. In order to avoid this it would be helpful if we could exclude several infeasible combinations of train types with the same inequality.

One promising idea is to detect a "comparably small" set of lines $\hat{\mathcal{R}} \subset \mathcal{R}$ which already causes the infeasibility of the present FSP instance. An approach to find such a set is given in section 4.6. Using $\hat{\mathcal{R}}$, the constraint (4.1) can be replaced by

$$\sum_{r \in \hat{\mathcal{R}}} x_{r,\tau_r} \leq |\hat{\mathcal{R}}| - 1. \tag{4.2}$$

It may even be allowed to exclude several train types for one line at the same time if these types have the same speed.

The ideas of this section are combined in algorithm 4.1.

## 4.4 Branch-and-Bound Method

A short overview on branch-and-bound methods and in particular for such methods for the solution of MIPs is given in appendix B. We will now focus on the important points for the design of a

**Algorithm 4.1** Relaxation Iteration Algorithm for the MCSP

**loop**
**if** the IP-MCTP is infeasible **then**
Stop. The MCSP is infeasible.
**end if**
Let the vector of optimal types for the IP-MCTP be given by $\boldsymbol{x}$ and the vector of optimal numbers of coaches given by $\boldsymbol{w}$.
**if** the FSP for $\boldsymbol{x}$ is feasible with solution vectors $\boldsymbol{a}, \boldsymbol{d}, \boldsymbol{z}$ **then**
Stop. An optimal solution is given by $\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{a}, \boldsymbol{d}, \boldsymbol{z}$.
**end if**
Let $\hat{\mathcal{R}}$ be a set of lines leading to the infeasibility of FSP for $\boldsymbol{x}$. Add inequality (4.2) for $\hat{\mathcal{R}}$ to the IP-MCTP.
**end loop**

branch-and-bound algorithm for the solution of the MCSP.

- *Choice of relaxation*: Like in section 4.3, the IP-MCTP relaxation is used. The remarks given in that section also apply to the branch-and-bound method.

- *Feasibility check*: Again this is done like in section 4.3.

- *Choice of division / partition*: If the FSP instance is infeasible, we need to change at least one of the train types. Let $\tau_r$ be the train type for line $r \in \mathcal{R}$, and let $\rho := |\mathcal{R}|$. An obvious way to divide the set $\mathcal{T}_1 \times \ldots \times \mathcal{T}_\rho$ of possible (but not necessarily feasible) combinations for train types is given by

$$\mathcal{T}_1 \times \ldots \times \mathcal{T}_\rho = (\mathcal{T}_1 \backslash \{\tau_1\}) \times \mathcal{T}_2 \times \ldots \times \mathcal{T}_\rho \cup \ldots \cup \mathcal{T}_1 \times \ldots \times \mathcal{T}_{\rho-1} \times (\mathcal{T}_\rho \backslash \{\tau_\rho\}). \tag{4.3}$$

  According to this scheme, $|\mathcal{R}|$ new problems have to be generated in this case. In section 4.3 we have suggested replacing $\mathcal{R}$ by a "small" set $\hat{\mathcal{R}} \subset \mathcal{R}$ which is already causing the conflict. This method can also be applied here.

By using these methods, a first version of a branch-and-bound algorithm for the MCSP is obtained, see algorithm 4.2. Again, we assume that $\mathcal{R} = \{r_1, \ldots, r_\rho\}$. Of course, if $i = 1$ the notation $\mathcal{T}_1 \times \ldots \times \mathcal{T}_{i-1} \times \mathcal{T}_i \backslash \{\tau_i\} \times \mathcal{T}_{i+1} \times \ldots \times \mathcal{T}_\rho$ means $\mathcal{T}_1 \backslash \{\tau_1\} \times \mathcal{T}_2 \times \ldots \times \mathcal{T}_\rho$ and is understood similarly for the other special cases.

We will now present several methods which accelerated algorithm 4.2 for our practical problem instances considerably:

- *LP relaxation for the IP-MCTP*: Instead of directly solving the IP-MCTP for $\mathcal{T}^l \in \mathcal{L}$, we only solve the LP relaxation of the IP-MCTP in a first step. If it is infeasible or has an objective value $\geq c_*$, i.e. worse than the value of the best known solution, the next branch-and-bound node can be examined immediately (without considering the IP).

**Algorithm 4.2** Simple Branch-and-Bound Algorithm for the MCSP

$c_* := \infty$; $\mathcal{L} := \{\{\mathcal{T}_1 \times \ldots \times \mathcal{T}_\rho\}\}$; $l_{\mathcal{T}_1 \times \ldots \times \mathcal{T}_\rho} := -\infty$
**loop**
  **if** $\mathcal{L} = \emptyset$ **then**
    Stop. If $c_* = \infty$, the problem is infeasible. Otherwise, an optimal solution is given by $\boldsymbol{x}_*$, $\boldsymbol{w}_*$, $\boldsymbol{a}_*$, $\boldsymbol{d}_*$, $\boldsymbol{z}_*$.
  **end if**
  Choose $\mathcal{T}' \in \mathcal{L}$.
  $\mathcal{L} := \mathcal{L} \backslash \{\mathcal{T}'\}$
  **if** the MCTP for $\mathcal{T}'$ is infeasible **then**
    **continue**
  **end if**
  Let an optimal solution of the IP-MCTP be defined by the vectors $\boldsymbol{x}$ and $\boldsymbol{w}$ with optimal value $c$.
  **if** $c \geq c_*$ **then**
    **continue**
  **end if**
  **if** the FSP for $\boldsymbol{x}$ is feasible with solution vectors $\boldsymbol{a}$, $\boldsymbol{d}$, $\boldsymbol{z}$ **then**
    $c_* := c$; $\boldsymbol{x}_* := \boldsymbol{x}$; $\boldsymbol{w}_* := \boldsymbol{w}$; $\boldsymbol{a}_* := \boldsymbol{a}$; $\boldsymbol{d}_* := \boldsymbol{d}$; $\boldsymbol{z}_* := \boldsymbol{z}$
    $\mathcal{L} := \mathcal{L} \backslash \{\hat{\mathcal{T}} \mid l_{\hat{\mathcal{T}}} \geq c_*\}$
    **continue**
  **end if**
  Let $\hat{\mathcal{R}}$ be a set of lines leading to the infeasibility of FSP for $\boldsymbol{x}$ and let $\tau_i$ be the train type for line $r_i \in \mathcal{R}$ in the MCTP solution defined by $\boldsymbol{x}$.
  **for** $i = 1$ to $\rho$ **do**
    **if** $r_i \in \hat{\mathcal{R}}$ **then**
      Let $\mathcal{T}^*$ denote $\mathcal{T}_1 \times \ldots \times \mathcal{T}_{i-1} \times \mathcal{T}_i \backslash \{\tau_i\} \times \mathcal{T}_{i+1} \times \ldots \times \mathcal{T}_\rho$.
      $l_{\mathcal{T}^*} := c$
      $\mathcal{L} := \mathcal{L} \cup \{\mathcal{T}^*\}$
    **end if**
  **end for**
**end loop**

- *Bound dominance*: Let $\mathcal{T}' := \mathcal{T}'_1 \times \ldots \times \mathcal{T}'_\rho$ and $\mathcal{T}'' := \mathcal{T}''_1 \times \ldots \times \mathcal{T}''_\rho$ be two possible (but not necessarily feasible) combinations of train types. If

$$\mathcal{T}' \supseteq \mathcal{T}'' \qquad \text{and} \qquad l_{\mathcal{T}'} \geq l_{\mathcal{T}''},$$

  we can set $l_{\mathcal{T}''} := l_{\mathcal{T}'}$, since the problem for $\mathcal{T}'$ is a relaxation of the problem for $\mathcal{T}''$.

  There are two situations in the algorithm where this dominance can be exploited. After the solution of the IP-MCTP (or even after the solution of the corresponding LP relaxation), the optimal value $c$ is a new lower bound for the problem defined by $\mathcal{T}'$. Assume that there is a set $\mathcal{T}'' \in \mathcal{L}$ with $\mathcal{T}'' \subset \mathcal{T}'$. Then this lower bound is also valid for $\mathcal{T}''$. If already $c \geq c_*$ holds, $\mathcal{T}''$ can be removed from $\mathcal{L}$ without ever being examined further. The same applies if the IP-MCTP for $\mathcal{T}'$ or the corresponding LP relaxation is infeasible.

  The second situation is the division of $\mathcal{T}'$. Sometimes, better bounds for the new problems can be obtained from elements already in $\mathcal{L}$.

These improvements lead to algorithm 4.3.

**Algorithm 4.3** Improved Branch-and-Bound Algorithm for the MCSP

$c_* := \infty$; $\mathcal{L} := \{\{\mathcal{T}_1 \times \ldots \times \mathcal{T}_\rho\}\}$; $l_{\mathcal{T}_1 \times \ldots \times \mathcal{T}_\rho} := -\infty$
**loop**
  **if** $\mathcal{L} = \emptyset$ **then**
    Stop. If $c_* = \infty$, the problem is infeasible. Otherwise, an optimal solution is given by $\boldsymbol{x}_*$, $\boldsymbol{w}_*$, $\boldsymbol{a}_*$, $\boldsymbol{d}_*$, $\boldsymbol{z}_*$.
  **end if**
  Choose $\mathcal{T}' \in \mathcal{L}$.
  $\mathcal{L} := \mathcal{L} \backslash \{\mathcal{T}'\}$
  **if** LP relaxation of IP-MCTP for $\mathcal{T}'$ is infeasible **then**
    $\mathcal{L} := \mathcal{L} \backslash \{\mathcal{T}'' \in \mathcal{L} \mid \mathcal{T}'' \subseteq \mathcal{T}'\}$;
    **continue**
  **end if**
  **if** LP relaxation of IP-MCTP has an optimal value $\geq c_*$ **then**
    $\mathcal{L} := \mathcal{L} \backslash \{\mathcal{T}'' \in \mathcal{L} \mid \mathcal{T}'' \subseteq \mathcal{T}'\}$;
    **continue**
  **end if**
  **if** the IP-MCTP for $\mathcal{T}'$ is infeasible **then**
    $\mathcal{L} := \mathcal{L} \backslash \{\mathcal{T}'' \in \mathcal{L} \mid \mathcal{T}'' \subseteq \mathcal{T}'\}$;
    **continue**
  **end if**
  Let an optimal solution of the IP-MCTP be defined by the vectors $\boldsymbol{x}$ and $\boldsymbol{w}$ with optimal value $c$.
  **if** $c \geq c_*$ **then**
    $\mathcal{L} := \mathcal{L} \backslash \{\mathcal{T}'' \in \mathcal{L} \mid \mathcal{T}'' \subseteq \mathcal{T}'\}$;
    **continue**
  **end if**
  **if** FSP for $\boldsymbol{x}$ is feasible with solution vectors $\boldsymbol{a}$, $\boldsymbol{d}$, $\boldsymbol{z}$ **then**
    $c_* := c$; $\boldsymbol{x}_* := \boldsymbol{x}$; $\boldsymbol{w}_* := \boldsymbol{w}$; $\boldsymbol{a}_* := \boldsymbol{a}$; $\boldsymbol{d}_* := \boldsymbol{d}$; $\boldsymbol{z}_* := \boldsymbol{z}$
    $\mathcal{L} := \mathcal{L} \backslash \{\hat{\mathcal{T}} \mid l_{\hat{\mathcal{T}}} \geq c_*\}$
    **continue**
  **end if**
  Let $\hat{\mathcal{R}}$ be a set of lines leading to the infeasibility of FSP for $\boldsymbol{x}$ and let $\tau_i$ be the train type for line $r_i \in \mathcal{R}$ in the MCTP solution defined by $\boldsymbol{x}$.
  **for** $i = 1$ to $\rho$ **do**
    **if** $r_i \in \hat{\mathcal{R}}$ **then**
      Let $\mathcal{T}^*$ denote $\mathcal{T}_1 \times \ldots \times \mathcal{T}_{i-1} \times \mathcal{T}_i \backslash \{\tau_i\} \times \mathcal{T}_{i+1} \times \ldots \times \mathcal{T}_\rho$.
      $l_{\mathcal{T}^*} := \max\{c, \max\{l_{\mathcal{T}''} \mid \mathcal{T}'' \in \mathcal{L} \wedge \mathcal{T}'' \supset \mathcal{T}^*\}\}$
      $\mathcal{L} := \mathcal{L} \cup \{\mathcal{T}^*\}$
    **end if**
  **end for**
**end loop**

As described in appendix B, in a branch-and-bound process one can use different node selection rules. For our implementation, we have used the following one: Always the node with the lowest

lower bound for the objective value is chosen. If there are several nodes with the same lowest bound, the one with the largest set $\mathcal{T}^*$ is used. By this selection, we hope that we obtain "good" lower bounds and our bounding strategies can be applied often.

## 4.5 Solving MCTP instances

In this section, solution methods for IP-MCTP instances are presented. A direct solution of the "IP-MCTP part" of the MIP-MCSP model from section 2.8 is possible for some instances with a commercial MIP solver, but it still takes too much time (recall that the decomposition algorithms may need to solve many such instances). In this section, we introduce a binary variable model for the MCTP, preprocessing techniques and cutting planes leading to a remarkable speedup of the solution process.

### Binary Variable Model

For the cost optimal line planning model, two integer linear formulations (COSTILP and COSTBLP, cf. section 2.7) have been examined. Our IP-MCTP model was developed from COSTILP. In an analogous way to COSTBLP, we can formulate a binary model for the MCTP. Therefore, we introduce the following variables:

$$w_{r,\tau,c} \quad \text{line } r \text{ uses train type } \tau \text{ with } c \text{ coaches}$$

The binary variable model *BP-MCTP* for the MCTP is given in figure 4.2. As one can see from table 4.1, the constraint matrix from the BP-MCTP has fewer rows, but more columns than the matrix for IP-MCTP. For our practical instances, the BP-MCTP provides better LP relaxations *and* shorter solution times. This experience is different from cost optimal line planning, where the binary formulation gave better LP relaxations and the general integer formulation gave better solution times. We may therefore replace IP-MCTP by BP-MCTP in our decomposition algorithms.

Binary variable model for MCTP (BP-MCTP):

$$\min \quad \sum_{r\in\mathcal{R}} \sum_{\tau\in\mathcal{T}_r} \sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} \left( \lceil \hat{t}_{r,\tau}/T \rceil \cdot (C^{\text{fix}}_\tau + c \cdot C^{\text{fixC}}_\tau) + d_r \cdot (C^{\text{km}}_\tau + c \cdot C^{\text{kmC}}_\tau) \right) w_{r,\tau,c}$$

$$\begin{aligned}
\sum_{r\in\mathcal{R}, r\ni e} \sum_{\tau\in\mathcal{T}_r} \sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} \mathfrak{C}_\tau \cdot c \cdot w_{r,\tau,c} &\geq N_e && \text{for each } e \in E \\
\sum_{\tau\in\mathcal{T}_r} \sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} w_{r,\tau,c} &= 1 && \text{for each } r \in \mathcal{R} \\
w_{r,\tau,c} &\in \{0,1\} && \text{for each } r \in \mathcal{R},\ \tau \in \mathcal{T}_r,\ c \in \{\underline{W}_\tau, \ldots, \overline{W}_\tau\}
\end{aligned}$$

Figure 4.2: Binary variable model for the MCTP

| Model | Number of rows | Number of columns | Number of nonzero entries |
|---|---|---|---|
| IP-MCTP | $\lvert\mathcal{R}\rvert + \lvert\mathcal{E}\rvert + 2\sum_{r\in\mathcal{R}} \lvert\mathcal{T}_r\rvert$ | $2\sum_{r\in\mathcal{R}} \lvert\mathcal{T}_r\rvert$ | $\sum_{e\in E}\sum_{\substack{r\in\mathcal{R}\\ r\ni e}} \lvert\mathcal{T}_r\rvert + 5\sum_{r\in\mathcal{R}} \lvert\mathcal{T}_r\rvert$ |
| BP-MCTP | $\lvert\mathcal{R}\rvert + \lvert\mathcal{E}\rvert$ | $\sum_{r\in\mathcal{R}}\sum_{\tau\in\mathcal{T}_r} 1+\overline{W}_\tau-\underline{W}_\tau$ | $\sum_{e\in E}\sum_{\substack{r\in\mathcal{R}\\ r\ni e}}\sum_{\tau\in\mathcal{T}_r} 1+\overline{W}_\tau-\underline{W}_\tau$ $+\sum_{r\in\mathcal{R}}\sum_{\tau\in\mathcal{T}_r} 1+\overline{W}_\tau-\underline{W}_\tau$ |

Table 4.1: Comparison of the models IP-MCTP and BP-MCTP

The following proposition shows that the optimal solution of the LP-relaxation for a BP-MCTP instance cannot be worse than the optimal solution of the relaxation for the corresponding IP-MCTP instance.

**Proposition 4.1** *Let $z^{\mathrm{I}}$ be the optimal solution value of the LP-relaxation of an IP-MCTP instance and let $z^{\mathrm{B}}$ be the optimal solution value of the LP-relaxation of the corresponding BP-MCTP instance. Then*

$$z^{\mathrm{I}} \leq z^{\mathrm{B}}.$$

**Proof:** We show that for each feasible solution $\boldsymbol{w}^{\mathrm{B}}$ of the BP-MCTP instance, there exists a feasible solution $(\boldsymbol{x}^{\mathrm{I}}, \boldsymbol{w}^{\mathrm{I}})$ of the IP-MCTP instance with the same objective value. Let $\boldsymbol{w}^{\mathrm{B}}$ be a feasible solution of the BP-MCTP instance. Define

$$x^{\mathrm{I}}_{r,\tau} := \sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} w^{\mathrm{B}}_{r,\tau,c} \qquad \text{and} \qquad w^{\mathrm{I}}_{r,\tau} := \sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} c\cdot w^{\mathrm{B}}_{r,\tau,c}.$$

One can easily verify that the obtained solution $(\boldsymbol{x}^{\mathrm{I}}, \boldsymbol{w}^{\mathrm{I}})$ satisfies all constraints of IP-MCTP. As an example, we consider the constraints for connecting $x$- and $w$-variables:

$$\underline{W}_\tau \cdot x_{r,\tau} = \underline{W}_\tau \cdot \sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} w_{r,\tau,c} \leq \sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} c\cdot w_{r,\tau,c} = w_{r,\tau} \leq \overline{W}_\tau \cdot \sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} w_{r,\tau,c} = \overline{W}_\tau \cdot x_{r,\tau}$$

The objective function values of $\boldsymbol{w}^{\mathrm{B}}$ and $(\boldsymbol{x}^{\mathrm{I}}, \boldsymbol{w}^{\mathrm{I}})$ are identical:

$$z^{\mathrm{I}} = \sum_{r\in\mathcal{R}}\sum_{\tau\in\mathcal{T}_r} \lceil \hat{t}_{r,\tau}/T\rceil \cdot (x_{r,\tau}\cdot C^{\mathrm{fix}}_\tau + w_{r,\tau}\cdot C^{\mathrm{fixC}}_\tau) + d_r\cdot(x_{r,\tau}\cdot C^{\mathrm{km}}_\tau + w_{r,\tau}\cdot C^{\mathrm{kmC}}_\tau) =$$

$$\sum_{r\in\mathcal{R}}\sum_{\tau\in\mathcal{T}_r}\left(\lceil \hat{t}_{r,\tau}/T\rceil \cdot \left(\left(\sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} w_{r,\tau,c}\right)\cdot C^{\mathrm{fix}}_\tau + \left(\sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} c\cdot w_{r,\tau,c}\right)\cdot C^{\mathrm{fixC}}_\tau\right)\right.$$

$$\left. + d_r\cdot\left(\left(\sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} w_{r,\tau,c}\right)\cdot C^{\mathrm{km}}_\tau + \left(\sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} c\cdot w_{r,\tau,c}\right)\cdot C^{\mathrm{kmC}}_\tau\right)\right) = z^{\mathrm{B}}$$

This completes the proof. □

### Preprocessing

By preprocessing the problem instances, we try to reduce their sizes or to improve their LP relaxations in order to accelerate the solution by our algorithms (cf. section B.3). For IP-MCTP instances (or BP-MCTP instances respectively), we will develop such preprocessing techniques now. The ideas are mainly based on combinatorial properties of the problem.

Often, one can find out in advance that on some network edge $e$, every feasible solution of the MCTP leads to a traveler capacity $N_e + \nu$, $\nu > 0$. In this case, $N_e$ can be increased by $\nu$ without changing the optimal solution, but possibly thereby obtaining a better LP relaxation value. Some situations where the traveler capacity can be increased are discussed now:

- *Greatest common divisor increase*: Let $e \in E$ and let $\Gamma_e$ be the greatest common divisor of all feasible coach capacities for trains serving the edge $e$, i.e.

$$\Gamma_e = \gcd\left\{ \mathfrak{C}_\tau \,\middle|\, \bigvee_{\substack{r \in \mathcal{R} \\ r \ni e}} \tau \in \mathcal{T}_r \right\}.$$

  Since every train running over $e$ has a capacity which is a multiple of $\Gamma_e$, the traveler capacity on every network edge $e$ can be modified in the following way:

$$N(e) := \left\lceil \frac{N(e)}{\Gamma_e} \right\rceil \cdot \Gamma_e$$

  This choice does not affect the feasibility of a solution of MCTP.

- *Line capacity bound increase*: Let $\underline{\mathfrak{C}}_r$ be a lower bound for the number of travelers carried by line $r$ in any feasible solution of MCTP. Then the following increase is valid:

$$N(e) := \max\left\{ N(e), \sum_{\substack{r \in \mathcal{R} \\ r \ni e}} \underline{\mathfrak{C}}_r \right\}$$

  A simple bound is $\underline{\mathfrak{C}}_r = \min\{\underline{W}_\tau \cdot \mathfrak{C}_\tau \mid \tau \in \mathcal{T}_r\}$. If there is a set of network edges $E' \subset E$ only served by $r$, possibly a better bound is given by

$$\underline{\mathfrak{C}}_r = \min\left\{ c \cdot \mathfrak{C}_\tau \,\middle|\, \tau \in \mathcal{T}_r,\ c \in \{\underline{W}_\tau, \ldots, \overline{W}_\tau\},\ c \cdot \mathfrak{C}_\tau \geq \max_{e \in E'} N(e) \right\}.$$

### Cutting Planes

We will now introduce cutting planes for BP-MCTP instances.

**Proposition 4.2** *Let $e \in E$ be a network edge and let $r_1, \ldots, r_n \in \mathcal{R}$ be the lines containing $e$. Let $n \geq 2$. Moreover, let $d_1, \ldots, d_{n-1} \in \{1, \ldots, N_e - 1\}$ with $d := \sum_{i=1}^{n-1} d_i < N_e$. Then*

$$\left( \sum_{i=1}^{n-1} \sum_{\tau \in \mathcal{T}_{r_i}} \sum_{\substack{c \in \{\underline{W}_\tau, \ldots, \overline{W}_\tau\} \\ c \cdot \mathfrak{C}_\tau \geq d_i}} w_{r_i,\tau,c} \right) + \sum_{\tau \in \mathcal{T}_{r_n}} \sum_{\substack{c \in \{\underline{W}_\tau, \ldots, \overline{W}_\tau\} \\ c \cdot \mathfrak{C}_\tau \geq N_e - d}} w_{r_n,\tau,c} \geq 1 \tag{4.4}$$

*is a valid inequality for the BP-MCTP.*

**Proof:** We assume the contrary. Because of the integrality, all variables appearing in (4.4) have to take a value of 0. It follows that

$$\sum_{\substack{r\in\mathcal{R}\\ r\ni e}}\sum_{\tau\in\mathcal{T}_r}\sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} c\cdot\mathfrak{C}_\tau\cdot w_{r,\tau,c} = \sum_{i=1}^{n}\sum_{\tau\in\mathcal{T}_{r_i}}\sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} c\cdot\mathfrak{C}_\tau\cdot w_{r_i,\tau,c} < \sum_{i=1}^{n-1} d_i + N_e - d = N_e,$$

which is a contradiction to the traveler capacity inequality of the BP-MCTP. □

For our practical instances, many of these cuts are violated by the respective LP relaxation, although the edges are mostly served only by a few lines. Even if $n = 2$, there are too many such violated inequalities to add them all. As the following proposition shows, for $n = 2$ it is only necessary to consider (4.4) for a few values of $d_1$.

**Proposition 4.3** *Let $e \in E$ be a network edge which is only served by the lines $r_1$ and $r_2$. Let $\mathfrak{C}_1^1,\ldots,\mathfrak{C}_1^{k_1}$ be all possible train capacities for trains of line $r_1$. Let $\mathfrak{C}_1^1 < \mathfrak{C}_1^2 < \ldots < \mathfrak{C}_1^{k_1}$, and let the respective values for $r_2$ be defined analogously.*

*Furthermore, let a solution for the LP relaxation of the BP-MCTP be given by the vector $\boldsymbol{w}$. Let $d_1 \in \{1,\ldots,N_e-1\}$ such that condition (4.4) is violated for $\boldsymbol{w}$.*

*Then there is also a $\hat{d_1} \in \{\mathfrak{C}_1^1+1,\ldots,\mathfrak{C}_1^{k_1}+1\}$ such that the condition is violated for $\boldsymbol{w}$.*

**Proof:** Note that $d_1 > \mathfrak{C}_1^1$ holds (otherwise (4.4) cannot be violated).

We consider two cases: Either $d_1 \geq \mathfrak{C}_1^{k_1}+1$ or there is in index $i \in \{1,\ldots,k_1-1\}$ such that $\mathfrak{C}_1^i+1 \leq d_1 \leq \mathfrak{C}_1^{i+1}$. In the first case, choose $\hat{d_1} := \mathfrak{C}_1^{k_1}+1$. We then obtain

$$\sum_{\tau\in\mathcal{T}_{r_1}}\sum_{\substack{c\in\{\underline{W}_\tau,\ldots,\overline{W}_\tau\}\\ c\cdot\mathfrak{C}_\tau\geq\hat{d_1}}} w_{r_1,\tau,c} + \sum_{\tau\in\mathcal{T}_{r_2}}\sum_{\substack{c\in\{\underline{W}_\tau,\ldots,\overline{W}_\tau\}\\ c\cdot\mathfrak{C}_\tau\geq N_e-\hat{d_1}}} w_{r_2,\tau,c} = \sum_{\tau\in\mathcal{T}_{r_2}}\sum_{\substack{c\in\{\underline{W}_\tau,\ldots,\overline{W}_\tau\}\\ c\cdot\mathfrak{C}_\tau\geq N_e-\hat{d_1}}} w_{r_2,\tau,c} \leq \sum_{\tau\in\mathcal{T}_{r_2}}\sum_{\substack{c\in\{\underline{W}_\tau,\ldots,\overline{W}_\tau\}\\ c\cdot\mathfrak{C}_\tau\geq N_e-d_1}} w_{r_2,\tau,c} < 1.$$

In other words, (4.4) is violated.

Otherwise, choose $\hat{d_1} := \mathfrak{C}_1^i+1$. Then it follows that

$$\sum_{\tau\in\mathcal{T}_{r_1}}\sum_{\substack{c\in\{\underline{W}_\tau,\ldots,\overline{W}_\tau\}\\ c\cdot\mathfrak{C}_\tau\geq\hat{d_1}}} w_{r_1,\tau,c} + \sum_{\tau\in\mathcal{T}_{r_2}}\sum_{\substack{c\in\{\underline{W}_\tau,\ldots,\overline{W}_\tau\}\\ c\cdot\mathfrak{C}_\tau\geq N_e-\hat{d_1}}} w_{r_2,\tau,c} = \sum_{\tau\in\mathcal{T}_{r_1}}\sum_{\substack{c\in\{\underline{W}_\tau,\ldots,\overline{W}_\tau\}\\ c\cdot\mathfrak{C}_\tau\geq d_1}} w_{r_1,\tau,c} + \sum_{\tau\in\mathcal{T}_{r_2}}\sum_{\substack{c\in\{\underline{W}_\tau,\ldots,\overline{W}_\tau\}\\ c\cdot\mathfrak{C}_\tau\geq N_e-\hat{d_1}}} w_{r_2,\tau,c} \leq$$

$$\sum_{\tau\in\mathcal{T}_{r_1}}\sum_{\substack{c\in\{\underline{W}_\tau,\ldots,\overline{W}_\tau\}\\ c\cdot\mathfrak{C}_\tau\geq d_1}} w_{r_1,\tau,c} + \sum_{\tau\in\mathcal{T}_{r_2}}\sum_{\substack{c\in\{\underline{W}_\tau,\ldots,\overline{W}_\tau\}\\ c\cdot\mathfrak{C}_\tau\geq N_e-d_1}} w_{r_2,\tau,c} < 1,$$

and again (4.4) is violated. This completes the proof. □

As a consequence, in the case of $n = 2$ only few inequalities have to be checked for violation. For larger values of $n$, the choice from proposition 4.3 can be used as a heuristic.

We will now analyze the quality of the cuts (4.4). Consider a graph consisting only of two nodes and a connecting edge $e$ with two lines $r_1$ and $r_2$ running over $e$. Let the possible train capacities (resulting from the combinations of train types and numbers of coaches) for line $r_1$ be given by $\mathfrak{C}_1^1,\ldots,\mathfrak{C}_1^{k_1}$ with $\mathfrak{C}_1^1 \leq \ldots \leq \mathfrak{C}_1^{k_1}$. In contrast to proposition 4.3, $\mathfrak{C}_1^i = \mathfrak{C}_1^{i+1}$ is possible for some $i \in \{1,\ldots,k_1-1\}$ if the same capacity can be obtained by selecting different combinations of train types and numbers of coaches. Let the respective values be defined for line $r_2$.

**Proposition 4.4** *With these definitions, the polyhedron P described by the constraints*

$$\sum_{\tau\in\mathcal{T}_{r_1}}\sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} w_{r_1,\tau,c}=1,\qquad \sum_{\tau\in\mathcal{T}_{r_2}}\sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} w_{r_2,\tau,c}=1,\qquad \boldsymbol{w}\geq 0$$

*and the constraints (4.4) for all values of $\hat{d}_1$ from proposition 4.3 is integral.*

**Proof:** We show that the constraint matrix is an interval matrix and thus totally unimodular (cf. [46]). Since the right hand sides of the constraints are integral, the proposition follows.

Let us order the columns of the constraint matrix as follows. Order the variables for line $r_1$ by increasing capacity $\mathfrak{C}_1^i$ and the variables for line $r_2$ by decreasing capacity. Now the constraint matrix looks like this (bound constraints are omitted):

| $\mathfrak{C}_1^1$ | $\mathfrak{C}_1^2$ | | … | | $\mathfrak{C}_1^{k_1}$ | $\mathfrak{C}_2^{k_2}$ | | … | | $\mathfrak{C}_2^2$ | $\mathfrak{C}_2^1$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | … | 1 | 1 | 0 | 0 | … | 0 | 0 | 0 | $\sum_\tau\sum_c w_{r_1,\tau,c}=1$ |
| 0 | 0 | 0 | … | 0 | 0 | 1 | 1 | … | 1 | 1 | 1 | $\sum_\tau\sum_c w_{r_2,\tau,c}=1$ |
| 0 | 1 | … | … | … | … | … | … | 1 | 0 | … | 0 | from proposition 4.3 |
| 0 | 0 | 1 | … | … | … | … | … | … | 1 | 0 | 0 | from proposition 4.3 |
| | | | | | etc. | | | | | | | from proposition 4.3 |

Obviously, it is an interval matrix. □

Now compare $P$ and the polyhedron $P'$ given by the LP relaxation of the corresponding BP-MCTP instance. Every integer point of $P$ satisfies the BP-MCTP traveler capacity constraints and thus is an element of $P'$. Conversely, all integer points from $P'$ are also in $P$: The constraints from proposition 4.4 are either the same as in the BP-MCTP description or are fulfilled because of proposition 4.2.

Since $P$ is integral, the addition of all cuts from proposition 4.3 is sufficient to obtain an integral polyhedron for the special graph we have examined. For general instances, we can conclude that if there is an edge with two lines running over it, there are no "better" cuts for BP-MCTP which are using only the information of the traveler volume on that edge and the available capacities for trains of the two lines running over that edge.

In [10], Bussieck introduces several classes of cutting planes for the line optimization model we have presented in section 2.7. One of these classes ((5.19)/(5.20) in [10]) can be adapted to the BP-MCTP. The following proposition deals with this class:

**Proposition 4.5** *Let $E'\subseteq E$, $N^{E'}:=\sum_{e\in E'}N_e$. Let $\alpha_r^{E'}$ denote the number of edges of $E'$ that are contained in line $r$. Let $\alpha^{E'}:=\max_{r\in\mathcal{R}}\alpha_r^{E'}$. Then the following inequality is valid:*

$$\sum_{\substack{r\in\mathcal{R}\\ \alpha_r^{E'}\geq 1}}\sum_{\tau\in\mathcal{T}_r}\sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} c\cdot\mathfrak{C}_\tau\cdot w_{r,\tau,c}\geq\left\lceil\frac{N^{E'}}{\alpha^{E'}}\right\rceil \tag{4.5}$$

**Proof:** For each $e \in E'$ we have

$$\sum_{\substack{r \in \mathcal{R} \\ r \ni e}} \sum_{\tau \in \mathcal{T}_r} \sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} c \cdot \mathfrak{C}_\tau \cdot w_{r,\tau,c} \geq N_e,$$

and therefore

$$\sum_{e \in E'} \sum_{\substack{r \in \mathcal{R} \\ r \ni e}} \sum_{\tau \in \mathcal{T}_r} \sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} c \cdot \mathfrak{C}_\tau \cdot w_{r,\tau,c} \geq N^{E'}.$$

Now

$$\alpha^{E'} \cdot \sum_{\substack{r \in \mathcal{R} \\ \alpha_r^{E'} \geq 1}} \sum_{\tau \in \mathcal{T}_r} \sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} c \cdot \mathfrak{C}_\tau \cdot w_{r,\tau,c} \geq \sum_{\substack{r \in \mathcal{R} \\ \alpha_r^{E'} \geq 1}} \sum_{\substack{e \in E' \\ r \ni e}} \sum_{\tau \in \mathcal{T}_r} \sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} c \cdot \mathfrak{C}_\tau \cdot w_{r,\tau,c} =$$

$$\sum_{e \in E'} \sum_{\substack{r \in \mathcal{R} \\ r \ni e}} \sum_{\tau \in \mathcal{T}_r} \sum_{c=\underline{W}_\tau}^{\overline{W}_\tau} c \cdot \mathfrak{C}_\tau \cdot w_{r,\tau,c} \geq N^{E'}.$$

Divide this inequality by $\alpha^{E'}$. Since the left hand side remains integral, the right hand side may be rounded up. We may even divide by $\alpha^{E'} \cdot \Gamma$, where $\Gamma$ is the greatest common divisor of all train capacities. □

The effect of the cutting planes (4.5) is visualized in figure 4.3. Assume that for each of the three lines given in the picture, there is only one train type, and let the capacity of one coach of this type be 10. Let the feasible numbers of coaches be 1 or 2.

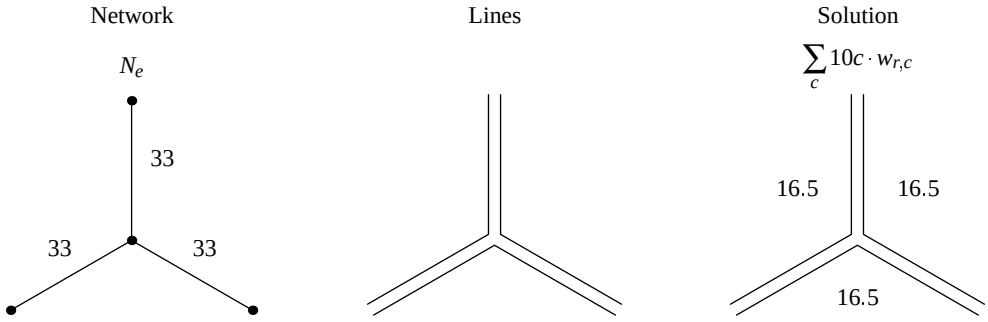



Figure 4.3: Solution without inequalities (4.5)

With (4.5) we obtain $\sum_{r,c} 10c \cdot w_{r,c} \geq 50$, which is violated by the solution.

For our practical instances, these cuts had only a very small effect on the LP relaxation value. In fact, they slowed the IP solution process down. Therefore, we finally have not used them.

## 4.6 Solving FSP instances

The problem of finding a feasible schedule for fixed train types can be formulated as satisfying a set of linear constraints with integer variables. A simple solution approach is adding an arbitrary linear

objective function and using a commercial MIP solver. For practical instances, this method takes too much time.

Since the structure of the FSP and the PESP is similar, another idea is to use a PESP algorithm and adapt it to the FSP. In chapter 3, several algorithms for solving PESP instances have been introduced. Most of them can be extended in order to handle FSP instances. We have chosen to adapt the PESP algorithm of Serafini and Ukovich (see section 3.6) for the FSP. This algorithm is fast enough for our instances and consumes only a small amount of memory.

We will also discuss a method for finding a "small" set of lines causing the conflict in case of infeasible instances.

### Modification of the Algorithm of Serafini and Ukovich

The PESP algorithm of Serafini and Ukovich has been introduced in section 3.6. Variants of this algorithm which lead to an acceleration for many practical instances have been discussed in section 3.7. In order to use the algorithm (or its variants) for FSP instances we will have to modify it in such a way that for every solution, the following two conditions are fulfilled (cf. section 2.5):

- The modulo parameters for travel time, waiting time and turning time constraints are 0.
- The modulo parameters for certain pairs of headway constraints are identical.

If there is no PESP solution satisfying these additional constraints, the FSP instance is infeasible. Of course, if the PESP instance already is infeasible, then so is the FSP instance.

We can ensure that the modulo parameters of the travel time, waiting time and turning time constraints are 0 by taking the corresponding arcs into the start tree of the algorithm of Serafini and Ukovich. This is possible because those arcs form a spanning forest of the event graph.

A naive idea to provide the equality of some modulo parameters in the algorithm of Serafini and Ukovich is to execute a backtracking step if this equality is violated somewhere in the search tree. This does not work correctly in general. An example is given in figure 4.4. For start tree 1, the algorithm states that no feasible solution exists, but for start tree 2, a feasible solution is found. The problem is that it is not possible to set the modulo parameter to 0 on an arbitrary spanning tree without changing the solution space (no statement analogous to proposition 3.2 exists).

For our FSP instances, we will show in proposition 4.6 that it is possible to select a start tree such that the algorithm gives the desired result: If there exists a solution for an instance with a modulo parameter of 0 on the travel time, waiting time and turning time arcs, together with equal modulo parameters on certain pairs of headway arcs, then there exists a solution with the additional property that on the chosen start tree, all modulo parameters are 0. This is possible because of the special structure of the FSP.

An example for an FSP event graph is given in figure 4.5. Here, the events are given upper indices $1, 2, \ldots$ in the order of appearance in the line circulations. $R_i$ is the set of all events belonging to line $r_i$. This notation is used in proposition 4.6.

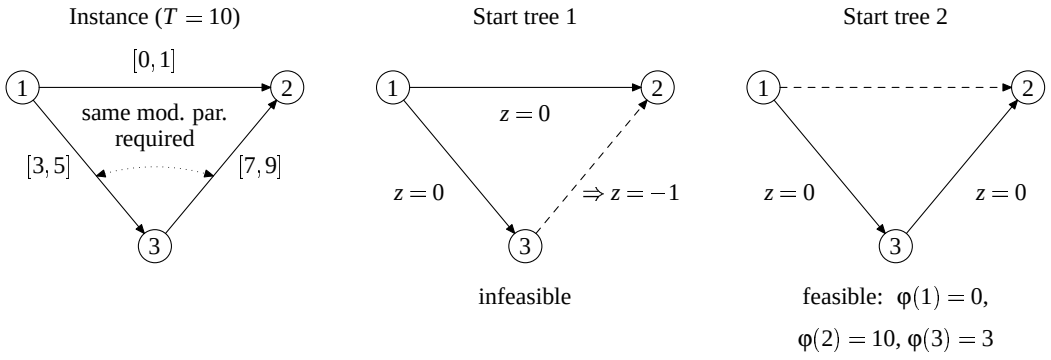



Figure 4.4: The naive algorithm may state feasibility only for some start trees.

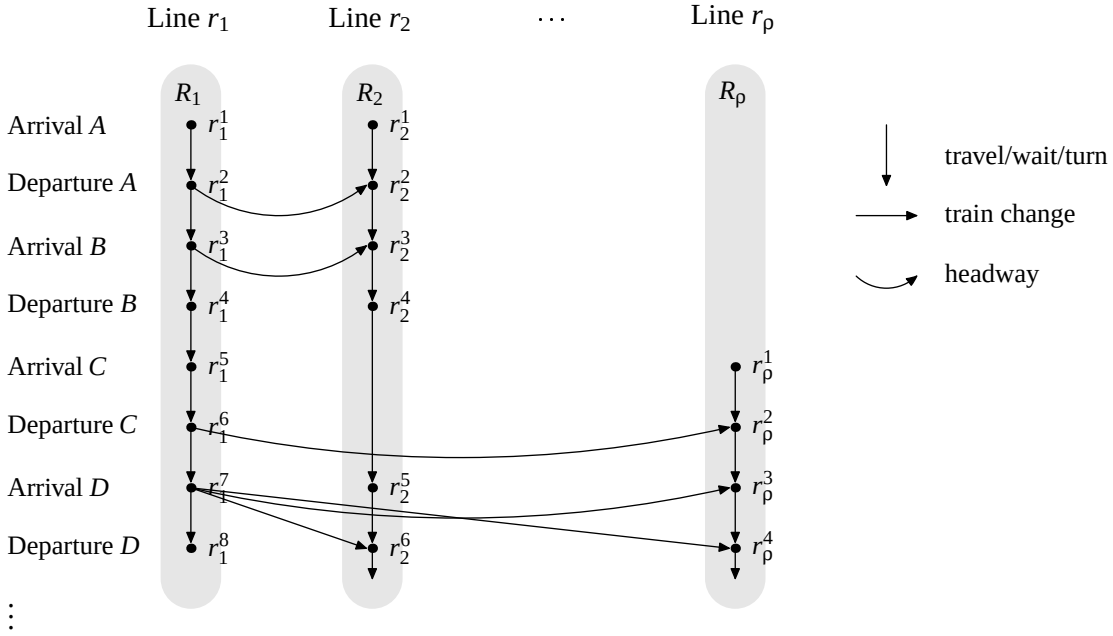



Figure 4.5: Example for an FSP event graph

**Proposition 4.6** *Consider a feasible FSP instance with event graph $\mathcal{G}$. Then, for each spanning tree $S$ of $\mathcal{G}$ that contains the travel time, waiting time and turning time arcs, there exists a feasible potential with a modulo parameter of* 0 *on the tree arcs.*

**Proof:** Let an FSP instance and a feasible potential $\varphi$ be given. Let $\mathcal{R} = \{r_1, \ldots, r_\rho\}$ and let the events of line $r_i$, $i \in \{1, \ldots, \rho\}$ be denoted by $r_i^1, r_i^2, \ldots, r_i^{k_i}$. $r_i^1$ is the arrival at the first station in direction $\mu = 0$, $r_i^2$ the corresponding departure etc., $r_i^{k_i}$ is the departure at the first station (i.e. the station of $r_i^1$) in direction $\mu = 1$, which is the last event of the line that has to be considered. Let $R_i := \{r_i^1, \ldots, r_i^{k_i}\}$ for each $i \in \{1, \ldots, \rho\}$.

Now we will construct a potential $\varphi'$ which is also feasible, but leads to a modulo parameter of 0 on all tree arcs. Note that by adding a multiple of the period to the potentials of all nodes in a

set $R_i$, $i \in \{1,\dots,\rho\}$, the modulo parameters of pairs of arcs needing the same modulo parameters are changed in the same way. Hence, one can add a multiple of the period to the potentials of such a set without violating the corresponding condition.

Now consider an arc $(r_i^j, r_{i'}^{j'})$ with $i,i' \in \{1,\dots,\rho\}$, $j \in \{1,\dots,k_i\}$, $j' \in \{1,\dots,k_{i'}\}$ with a nonzero modulo parameter $z$ with respect to $\varphi$. Subtract $z \cdot T$ from the potentials of all nodes $v \in V_{\mathcal{G}}$ with a path (arc direction is ignored here) from $r_{i'}^{j'}$ to $v$ containing only arcs from $S \backslash (r_i^j, r_{i'}^{j'})$. This procedure changes the modulo parameter only on one tree arc, namely $(r_i^j, r_{i'}^{j'})$. In fact, the nodes $v$ are exactly the members of a union of some $R_i$, $i \in \{1,\dots,\rho\}$.

By applying this method, the modulo parameter of the arc $(r_i^j, r_{i'}^{j'})$ is set to 0, while the modulo parameters of the other arcs of $S$ remain unchanged. Further, only modulo parameters of complete sets $R_i$, $i \in \{1,\dots,\rho\}$ are changed, and so the modulo parameters of arcs needing the same parameter are always changed in the same way. Thus, we can iteratively construct a potential $\varphi$ with a modulo parameter of 0 on all tree arcs. □

This proposition allows us to use the algorithm of Serafini and Ukovich to solve FSP instances. We only need to select all travel time, waiting time and turning time constraints for the start tree. If the modulo parameters for arcs needing the same value are chosen differently, a backtracking step can be executed.

### Finding a Set of Lines Causing Infeasibility

As we have already pointed out, it would be helpful for our algorithms if we could not only detect that an FSP instance is infeasible, but also could determine which lines actually cause the conflict.

Look at the example of figure 4.6. Suppose that after constructing the start tree, the algorithm of Serafini and Ukovich tries to satisfy the interval condition for the dashed arc and fails (infeasibility). If some travel time intervals are changed in $R_2$ for example, and the algorithm is restarted, then the start tree is constructed from the same arcs again (which will be the case when the algorithm is executed as suggested), the same algorithmic steps will be performed. The dashed arc again cannot be satisfied. Therefore, changing a train type on line $r_2$ will not resolve the conflict. This argumentation will be formalized in the following proposition.

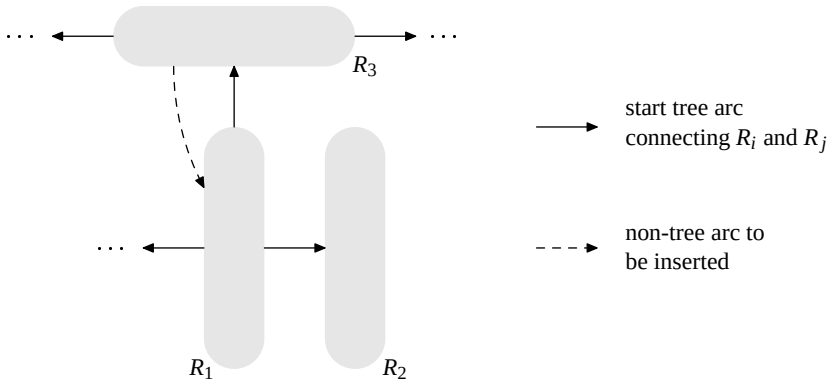



Figure 4.6: FSP start tree and a new arc

**Proposition 4.7** *Let the start tree S for the FSP version of the algorithm of Serafini and Ukovich always be chosen independently of the train types. Let the order in which non-tree arcs are examined be independent of the train types. Let there be an infeasible FSP instance where the infeasibility has been detected after examining the non-tree arcs $a_1, \dots, a_q \in A_{\mathcal{G}}$. Let $Q_i$ be the node set of the cycle containing $a_i$ and the path from the end node of $a_i$ to the start node using only arcs from S.*

*Then the FSP instance remains infeasible if only train types for lines with nodes not contained in $\bigcup_{i=1}^{q} Q_i$ are changed.*

**Proof:** In this case, the algorithm performs the same steps. □

With this proposition, we can define

$$\hat{\mathcal{R}} := \left\{ r \in \mathcal{R} \;\middle|\; \bigvee_{i \in \{1,\dots,q\}} r \text{ contains a node of } Q_i \right\}$$

for our relaxation iteration or branch-and-bound algorithm.

## 4.7 Exact Solution of the Nonlinear Problem

In section 2.7, we introduced the estimation $\hat{t}_r$ for the circulation time of line $r$ in order to calculate the number of trains required for the operation of the line in the line planning models. The estimation was also used in the schedule optimization model.

The actual circulation time $t_r$ for a line $r = (v_1, \dots, v_n) \in \mathcal{R}$ depends on the schedule. If we assume that the minimum time for turning from direction $\mu = 1$ to direction $\mu = 0$ is used, $t_r$ is given by

$$t_r = d_{r,1}^{v_1} - a_{r,0}^{v_1} + \underline{turn}. \tag{4.6}$$

Let N-MCSP be the *nonlinear* model obtained from MIP-MCSP by replacing $\hat{t}_{r,\tau}$ by $t_{r,\tau}$ and adding the constraints (4.6) for each line. We will now construct an algorithm for solving N-MCSP exactly. Therefore, we will modify the FSP algorithm so that we can find out the number of trains used in a feasible solution and the corresponding cost. After this modification, we will adapt the relaxation iteration and the branch-and-bound method.

### Determining Cost with the FSP Algorithm

The number of trains required for a line and thus the cost of a solution of an FSP instance can be derived from certain modulo parameters of an extended event graph. We will now again distinguish between events and event times. The number of trains needed for a line $r = (v_1, \dots, v_n) \in \mathcal{R}$ is given by

$$\gamma_r := \left\lceil \frac{\pi(d_{r,1}^{v_1}) - \pi(a_{r,0}^{v_1}) - \underline{turn}}{T} \right\rceil,$$

where we use the minimum turning time, because our objective is to minimize the number of trains.

Now insert two arcs for each line $r = (v_1, \dots, v_n) \in \mathcal{R}$:

- $c_r^{\text{end}} := (d_{r,1}^{v_1}, a_r^{\text{end}}, \underline{turn} + T - 1, \underline{turn} + T - 1)$ (always use this arc in the start tree)
- $c_r^{\text{loop}} := (a_{r,0}^{v_1}, a_r^{\text{end}}, 0, T - 1)$

The arcs $c_r^{\text{loop}}$ for each $r \in \mathcal{R}$ are always feasible. Now the modulo parameter on an arc $c_r^{\text{loop}}$ is exactly the number of trains needed for line $r$. In order to verify this, we consider two cases:

- $\pi(d_{r,1}^{v_1}) - \pi(a_{r,0}^{v_1}) + \underline{turn} = k \cdot T$ for $k \in \mathbb{N}$. In this case, $\gamma_r = k$. Now consider the modulo parameter for the arc $c_r^{\text{loop}}$:

$$\begin{array}{rcccl} 0 & \leq & \pi(a_r^{\text{end}}) - \pi(a_{r,0}^{v_1}) - z \cdot T & \leq & T-1 \qquad z \in \mathbb{Z} \\ 0 & \leq & \pi(d_{r,1}^{v_1}) - \pi(a_{r,0}^{v_1}) + \underline{turn} + T - 1 - z \cdot T & \leq & T-1 \\ 0 & \leq & k \cdot T + T - 1 - z \cdot T & \leq & T-1 \qquad \Rightarrow z = k \end{array}$$

  Recall that representative trains are used and thus the $\pi$-variables correspond to the same train.

- Now let $\pi(d_{r,1}^{v_1}) - \pi(a_{r,0}^{v_1}) + \underline{turn} = k \cdot T + t$ with $k \in \mathbb{N}$, $t \in \{1, \ldots, T-1\}$. We obtain $\gamma_r = k + 1$. In an analogous way

$$0 \quad \leq \quad k \cdot T + t + T - 1 - z \cdot T \quad \leq \quad T - 1 \qquad z \in \mathbb{Z},$$

  which can only be true for $z = k + 1$.

Let $\tau_r$ be the fixed train type of trains of line $r$ and let $w_r$ be the fixed number of coaches of trains of line $r$. Then the fixed cost part of an FSP solution with modulo parameter vector $\boldsymbol{z}$ is given by

$$\sum_{r \in \mathcal{R}} z_{c_r^{\text{loop}}} \cdot \left( C_{\tau_r}^{\text{fix}} + w_r \cdot C_{\tau_r}^{\text{fixC}} \right). \tag{4.7}$$

The km-oriented cost part is still independent of the schedule. Note that not only the train types, but also the number of coaches influences the cost of the schedule.

The FSP version of the algorithm of Serafini and Ukovich stops as soon as a feasible solution is found. Instead, we now evaluate (4.7) and execute a backtracking step. By this procedure, the complete search tree is examined. If there are no more possible modulo parameters, the algorithm terminates. If there are feasible solutions, an optimal solution is returned. Otherwise, infeasibility is stated.

We can reduce the number of search tree nodes that have to be visited by several techniques:

- In every node, we can calculate a lower bound on each modulo parameter (cf. chapter 3) and therefore a lower bound on the cost of the schedule. If this bound exceeds the value of the best known solution, the subtree for the node does not need to be examined.
- As a heuristic, when branching on an arc $c_r^{\text{loop}}$ we can always choose the subtree with the lower modulo parameter for $c_r^{\text{loop}}$ first, hoping to find a low cost solution early. With such a solution, it may be easier to find subtrees with a lower bound exceeding the best known cost bound.
- We may heuristically branch on arcs $c_r^{\text{loop}}$ early in the search tree in order to get strong lower bounds.

### Exact Relaxation Iteration Method

We will now extend algorithm 4.1 from section 4.3 in order to get an exact solution method for the N-MCSP. As a relaxation, BP-MCTP is used instead of IP-MCTP. On reason for doing this are the shorter solution times. Another reason is given below. There are two main differences between algorithm 4.1 and the exact method given in this section:

- The new algorithm does not stop as soon as an FSP instance is feasible, but cuts off the one combination of train types and numbers of coaches from the BP-MCTP that led to the instance:

  Let $\tau_r$ be the train type and $c_r$ the number of coaches of line $r \in \mathcal{R}$ in the BP-MCTP solution that led to the FSP instance. Then the inequality

  $$\sum_{r \in \mathcal{R}} w_{r,\tau_r,c_r} \leq |\mathcal{R}| - 1 \tag{4.8}$$

  gives the desired result for the model BP-MCTP. For the general integer model, there is no such simple inequality. This is the second advantage of using BP-MCTP instead of IP-MCTP.

  Inequality (4.2) can be easily transferred to the BP-MCTP. Let $\tau_r$ be the train type of the BP-MCTP solution. The corresponding binary model inequality is given by

  $$\sum_{r \in \hat{\mathcal{R}}} \sum_{c=\underline{W}_{\tau_r}}^{\overline{W}_{\tau_r}} w_{r,\tau_r,c} \leq |\hat{\mathcal{R}}| - 1. \tag{4.9}$$

- For the relaxation BP-MCTP, lower bounds on the line circulation time $\underline{t}_r$ are used instead of estimations $\hat{t}_r$. The lower bounds are calculated by always choosing the lower bound for travel, waiting and turning time.

Algorithm 4.4 is an exact relaxation iteration method for solving N-MCSP. It is based on the binary formulation BP-MCTP, thus there is no train type vector $\boldsymbol{x}$.

Practical experiences show that this method is very slow. In the worst case, $\prod_{r \in \mathcal{R}} \prod_{\tau \in \mathcal{T}_r} (1 + \overline{W}_\tau - \underline{W}_\tau)$ iterations are necessary. We have implemented a version of the algorithm only considering the train types for the cuts, i.e. inequality (4.8) is replaced by

$$\sum_{r \in \mathcal{R}} \sum_{c=\underline{W}_{\tau_r}}^{\overline{W}_{\tau_r}} w_{r,\tau_r,c} \leq |\mathcal{R}| - 1.$$

This inexact version is still much too slow (cf. chapter 5).

### Exact Branch-and-Bound Method

It is also possible to design an exact branch-and-bound method for the N-MCTP. As in the case of the exact relaxation iteration algorithm, we use BP-MCTP with lower bounds on the circulation time as a relaxation. In order to consider the number of coaches for each FSP instance, we extend the data for a subproblem.

**Algorithm 4.4** Exact Relaxation Iteration Algorithm for the N-MCSP

$c_* = \infty$
**loop**
  **if** the BP-MCTP is infeasible **then**
    Stop. If $c_* = \infty$, the problem is infeasible. Otherwise, an optimal solution with value $c_*$ is given by $\boldsymbol{w}_*$, $\boldsymbol{a}_*$, $\boldsymbol{d}_*$, $\boldsymbol{z}_*$.
  **end if**
  Let $\boldsymbol{w}$ be an optimal solution vector of the BP-MCTP (using $\underline{t}_r$) with objective value $c$.
  **if** $c \geq c_*$ **then**
    Stop. If $c_* = \infty$, the problem is infeasible. Otherwise, an optimal solution with value $c_*$ is given by $\boldsymbol{w}_*$, $\boldsymbol{a}_*$, $\boldsymbol{d}_*$, $\boldsymbol{z}_*$.
  **end if**
  **if** the FSP for $\boldsymbol{w}$ is feasible **then**
    Let an optimal solution for the FSP concerning (4.7) be given by $\boldsymbol{a}$, $\boldsymbol{d}$, $\boldsymbol{z}$. Let the objective value be $c$.
    **if** $c < c_*$ **then**
      $c_* := c$; $\boldsymbol{w}_* := \boldsymbol{w}$; $\boldsymbol{a}_* := \boldsymbol{a}$; $\boldsymbol{d}_* := \boldsymbol{d}$; $\boldsymbol{z}_* := \boldsymbol{z}$
    **end if**
    Add cut (4.8) to the BP-MCTP.
  **else**
    Let $\hat{\mathcal{R}}$ be a set of lines leading to the infeasibility of the FSP for $\boldsymbol{w}$.
    Add cut (4.9) to the BP-MCTP.
  **end if**
**end loop**

Let the feasible combinations of train types and numbers of coaches for a line $r \in \mathcal{R}$ be defined by

$$\mathcal{T}_r^c := \left\{ (\tau, c) \;\middle|\; \tau \in \mathcal{T}_r,\ c \in \{\underline{W}_\tau, \ldots, \overline{W}_\tau\} \right\}.$$

If $\mathcal{R} = \{r_1, \ldots, r_\rho\}$, then a subproblem is defined by $\mathcal{T}_1^c \times \ldots \times \mathcal{T}_\rho^c$.

The exact branch-and-bound method for the N-MCTP is given by algorithm 4.5. Again, this method is too slow for practical instances, and even a version considering only train types for generating new subproblems does not seem to be promising (see computational results in chapter 5).

**Algorithm 4.5** Exact Branch-and-Bound Algorithm for the N-MCSP

$c_* := \infty$; $\mathcal{L} := \{\{\mathcal{T}_1^c \times \ldots \times \mathcal{T}_\rho^c\}\}$; $l_{\mathcal{T}_1^c \times \ldots \times \mathcal{T}_\rho^c} := -\infty$
**loop**
  **if** $\mathcal{L} = \emptyset$ **then**
    Stop. If $c_* = \infty$, the problem is infeasible. Otherwise, ($\boldsymbol{w}_*$, $\boldsymbol{a}_*$, $\boldsymbol{d}_*$, $\boldsymbol{z}_*$) is optimal.
  **end if**
  Choose $\mathcal{T}' \in \mathcal{L}$.
  $\mathcal{L} := \mathcal{L} \backslash \{\mathcal{T}'\}$
  **if** LP relaxation of the BP-MCTP for $\mathcal{T}'$ is infeasible **then**
    $\mathcal{L} := \mathcal{L} \backslash \{\mathcal{T}'' \in \mathcal{L} \mid \mathcal{T}'' \subseteq \mathcal{T}'\}$; **continue**
  **end if**
  **if** LP relaxation of BP-MCTP has an optimal value $\geq c_*$ **then**
    $\mathcal{L} := \mathcal{L} \backslash \{\mathcal{T}'' \in \mathcal{L} \mid \mathcal{T}'' \subseteq \mathcal{T}'\}$; **continue**
  **end if**
  **if** the BP-MCTP for $\mathcal{T}'$ is infeasible **then**
    $\mathcal{L} := \mathcal{L} \backslash \{\mathcal{T}'' \in \mathcal{L} \mid \mathcal{T}'' \subseteq \mathcal{T}'\}$; **continue**
  **end if**
  Let an optimal solution of BP-MCTP be defined by the vector $\boldsymbol{w}$ with optimal value $c$.
  **if** $c \geq c_*$ **then**
    $\mathcal{L} := \mathcal{L} \backslash \{\mathcal{T}'' \in \mathcal{L} \mid \mathcal{T}'' \subseteq \mathcal{T}'\}$; **continue**
  **end if**
  **if** FSP for $\boldsymbol{w}$ is feasible **then**
    Let an optimal solution for the FSP concerning (4.7) be given by $\boldsymbol{a}$, $\boldsymbol{d}$, $\boldsymbol{z}$ with objective value $c'$.
    **if** $c' < c_*$ **then**
      $c_* := c'$; $\boldsymbol{w}_* := \boldsymbol{w}$; $\boldsymbol{a}_* := \boldsymbol{a}$; $\boldsymbol{d}_* := \boldsymbol{d}$; $\boldsymbol{z}_* := \boldsymbol{z}$; $\mathcal{L} := \mathcal{L} \backslash \{\hat{\mathcal{T}} \mid l_{\hat{\mathcal{T}}} \geq c_*\}$
    **end if**
    Let $\tau_i$ be the train type and $q_i$ the number of coaches for line $r_i \in \mathcal{R}$ in the BP-MCTP solution defined by $\boldsymbol{w}$.
    **for** $i = 1$ to $\rho$ **do**
      Let $\mathcal{T}^*$ denote $\mathcal{T}_1^c \times \ldots \times \mathcal{T}_{i-1}^c \times \mathcal{T}_i^c \backslash \{(\tau_i, q_i)\} \times \mathcal{T}_{i+1}^c \times \ldots \times \mathcal{T}_\rho^c$.
      $l_{\mathcal{T}^*} := \max\{c, \max\{l_{\mathcal{T}''} \mid \mathcal{T}'' \in \mathcal{L} \wedge \mathcal{T}'' \supset \mathcal{T}^*\}\}$
      $\mathcal{L} := \mathcal{L} \cup \{\mathcal{T}^*\}$
    **end for**
    **continue**
  **end if**
  Let $\hat{\mathcal{R}}$ be a set of lines leading to the infeasibility of the FSP for $\boldsymbol{w}$ and let $\tau_i$ be the train type for line $r_i \in \mathcal{R}$ in the BP-MCTP solution defined by $\boldsymbol{w}$.
  **for** $i = 1$ to $\rho$ **do**
    **if** $r_i \in \hat{\mathcal{R}}$ **then**
      Let $\mathcal{T}^*$ denote $\mathcal{T}_1^c \times \ldots \times \mathcal{T}_{i-1}^c \times \mathcal{T}_i^c \backslash \{(\tau_i, q) \mid q \in \{\underline{W}_{\tau_i}, \ldots, \overline{W}_{\tau_i}\}\} \times \mathcal{T}_{i+1}^c \times \ldots \times \mathcal{T}_\rho^c$.
      $l_{\mathcal{T}^*} := \max\{c, \max\{l_{\mathcal{T}''} \mid \mathcal{T}'' \in \mathcal{L} \wedge \mathcal{T}'' \supset \mathcal{T}^*\}\}$
      $\mathcal{L} := \mathcal{L} \cup \{\mathcal{T}^*\}$
    **end if**
  **end for**
**end loop**

# Chapter 5

# Computational Results

In this chapter, we report on computational experiences with the models PESP and MCSP introduced in the previous chapters. We have tested several algorithms for data from the railroad companies of Germany and the Netherlands. In section 5.1, we shortly describe the test instances. The other sections contain computational results for PESP instances (section 5.3) and MCSP instances (section 5.4).

## 5.1 Test Instances

We have tested our algorithms on real networks from the German railroad company *Deutsche Bahn (DB)* and the railroad company of the Netherlands *Nederlandse Spoorwegen (NS)*. For the German railroad, we used network data, line plans, origin destination matrices and cost data for the InterCity (IC) and InterRegio (IR) supply networks. For the railroad of the Netherlands, we obtained the respective data for the InterRegio (IR), InterCity (IC) and AggloRegio (AR) supply networks.

Furthermore, the PESP algorithms were tested on 15 special instances we obtained from NS. The constraint sets of these instances contain a subset of so called *marketing constraints* $\mathcal{C}^{\mathrm{M}} \subset \mathcal{C}$. These constraints are not absolutely necessary, but try to make the timetable attractive for passengers. Examples for those constraints are train change time constraints or constraints ensuring that lines running on the same track have a very large headway in order to get a short waiting time for passengers wishing to travel with an arbitrary of those lines. The instances obtained from instances 1–15 by ignoring the marketing constraints are called 1a–15a.

We start with a short characterization of the 30 resulting PESP instances. The numbers of nodes and arcs of the instances are given in table 5.1.

Some characteristics of the optimization instances can be found in table 5.2. For all instances, 4 different train types have been considered. A heuristic method for the determination of stations and lines used by passengers for changing trains is discussed shortly during the analysis of the optimization results.

As an example, the InterCity network of the Netherlands is given in figure 5.1. Cost optimal lines for this network and for the other networks of the Netherlands have been determined in [10].

| Inst. | # Nodes | # Arcs | Inst. | # Nodes | # Arcs | Inst. | # Nodes | # Arcs |
|---|---|---|---|---|---|---|---|---|
| 1 | 1866 | 14205 | 11 | 536 | 4705 | 6a | 1344 | 7477 |
| 2 | 1672 | 14707 | 12 | 265 | 1491 | 7a | 2338 | 12725 |
| 3 | 1672 | 11331 | 13 | 2233 | 14183 | 8a | 2338 | 12725 |
| 4 | 125 | 925 | 14 | 2395 | 14446 | 9a | 2338 | 12725 |
| 5 | 197 | 1118 | 15 | 2621 | 13175 | 10a | 2338 | 12725 |
| 6 | 1345 | 9443 | 1a | 1866 | 12967 | 11a | 536 | 4318 |
| 7 | 2339 | 13906 | 2a | 1596 | 11010 | 12a | 264 | 1259 |
| 8 | 2339 | 13924 | 3a | 1596 | 9752 | 13a | 2224 | 11925 |
| 9 | 2339 | 14264 | 4a | 124 | 721 | 14a | 2338 | 12717 |
| 10 | 2339 | 14102 | 5a | 196 | 920 | 15a | 2621 | 12953 |

Table 5.1: Numbers of nodes and arcs for the PESP instances

| | DB-IC | DB-IR | NS-IC | NS-IR | NS-AR |
|---|---|---|---|---|---|
| # Nodes | 90 | 297 | 36 | 38 | 122 |
| # Edges | 107 | 384 | 48 | 40 | 134 |
| # Lines | 31 | 89 | 25 | 21 | 117 |
| Average # edges in a line | 7.5 | 5.9 | 5.0 | 5.8 | 4.2 |

Table 5.2: Problem characteristics for the 5 optimization instances

## 5.2 Hardware and Software

Our computational experiments have been performed on a 400 MHz Pentium II PC with 256 MB main memory and operating system Linux.

The algorithms have been coded in C. For the solutions of MIPs and LPs, we used the commercial solver CPLEX, version 6.5. Details on this solver can be found in [34].

## 5.3 PESP Results

In this section we present and discuss experiences with several PESP algorithms, i.e. algorithms for finding a feasible schedule. We have implemented the PESP preprocessing methods from section 3.1 (without the decomposition methods, which could not be applied to our instances). Applying the preprocessing leads to a remarkable reduction of the event graph sizes of our set of instances, see table 5.3.

The preprocessing of each instance required always less than one minute.

In chapter 3, different algorithms for solving PESP instances were discussed. We have tested several of these on our instances. The obtained results can be found in table 5.4, where the columns correspond to different algorithms:

- MIP: We have solved instances by using CPLEX on the MIP formulation of the PESP. The

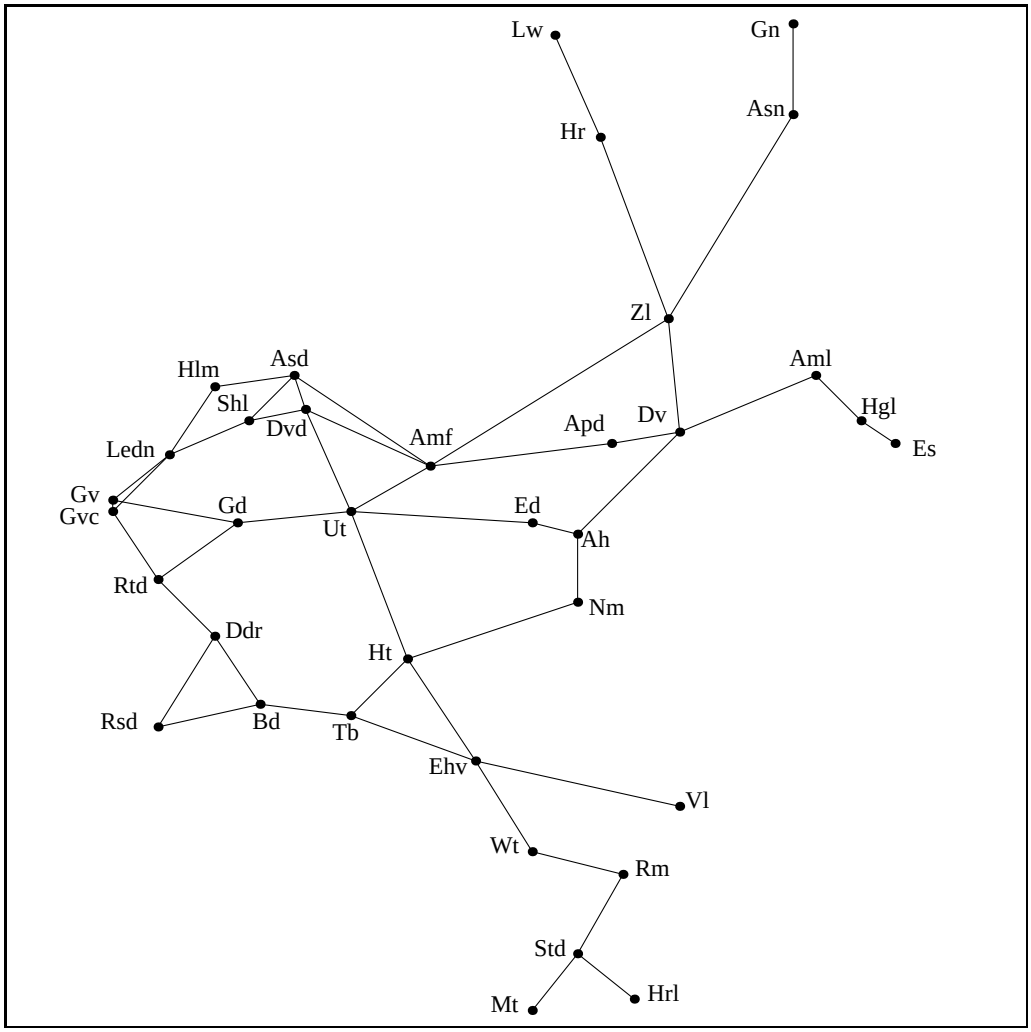



Figure 5.1: InterCity network of the Netherlands

solution times depend on the parameter settings of CPLEX. We obtained the best results by changing only a few parameters from their default values: We used *strong branching*, automatic generation of *fractional cuts* and an *aggregator tolerance* of $10^{-6}$. For a detailed explanation of these parameters, we refer to [5, 34].

- SU: This is the original algorithm of Serafini and Ukovich with the correction of Nachtigall (cf. section 3.6).
- SU†: This is the algorithm of Serafini and Ukovich, but with an arc preorder by the number of feasible modulo parameters (cf. section 3.7)
- SU‡: We have examined the Serafini-Ukovich algorithm with a dynamic strategy for the choice of arcs. In section 3.7, we have discussed several such strategies. In table 5.4, the fastest solution time obtained by a certain combination of these strategies is given. A detailed analysis of the strategies will be presented below.

| I | Nodes/Arcs original | Nodes/Arcs preproc. | I | Nodes/Arcs original | Nodes/Arcs preproc. | I | Nodes/Arcs original | Nodes/Arcs preproc. |
|---|---|---|---|---|---|---|---|---|
| 1 | 1866/14205 | 450/2975 | 11 | 536/4705 | 234/1430 | 6a | 1344/7477 | 1327/7157 |
| 2 | 1672/14707 | 1619/14404 | 12 | 265/1491 | 197/1056 | 7a | 2338/12725 | 1340/7823 |
| 3 | 1672/11331 | 515/7924 | 13 | 2233/14813 | 1098/7540 | 8a | 2338/12725 | 1340/7823 |
| 4 | 125/925 | 118/715 | 14 | 2395/14446 | 1287/8310 | 9a | 2338/12725 | 1340/7821 |
| 5 | 197/1118 | 182/951 | 15 | 2621/13175 | 446/1915 | 10a | 2338/12725 | 1340/7819 |
| 6 | 1345/9443 | 1096/6665 | 1a | 1866/12967 | 828/4829 | 11a | 536/4318 | 422/2132 |
| 7 | 2339/13906 | 1247/7923 | 2a | 1596/11010 | 706/3790 | 12a | 264/1259 | 218/1147 |
| 8 | 2339/13924 | 1247/7937 | 3a | 1596/9752 | 706/3655 | 13a | 2224/11925 | 1285/7254 |
| 9 | 2339/14264 | 1247/8141 | 4a | 124/721 | 123/694 | 14a | 2338/12717 | 1340/7811 |
| 10 | 2339/14102 | 1247/8050 | 5a | 196/920 | 174/862 | 15a | 2621/12953 | 446/1842 |

Table 5.3: Reduction of the event graph size by preprocessing

- BC: Results for the branch-and-cut algorithm from section 3.9) are given here.

The $*$-symbol in table 5.4 in the following tables indicates that there was no result after the limit of 10 hours of computation time. For some instances, it is actually unknown whether they are feasible or infeasible. This is indicated by a question mark in the corresponding column. There are some instances which could be solved (or proven to be infeasible), but with a solution time of much more than 10 hours (e.g. several days).

For some instances, the algorithm of Serafini and Ukovich detects infeasibility before building the search tree (0 nodes required). In this case, the instance is trivially infeasible (cf. section 3.1).

A detailed examination of the effects of arc choice strategies and heuristics of section 3.7 is given in table 5.5, where the following arc choice rules have been considered:

- A: arc with minimal number of feasible modulo parameters; arc search is terminated as soon as an arc with less than 2 feasible modulo parameters is found
- B: as A, but the arcs are examined in a cyclic way
- C: as B, but if there are several arcs with the minimum number of feasible modulo parameters $\geq 2$, the arc with maximal look-ahead value is chosen (cf. section 3.7); 5 arcs with the minimum number of feasible modulo parameters are examined
- D: as C, but more than 5 arcs are examined as long as the product of the number of examined arcs and the best look-ahead value is $\leq 100$
- E: as D, but arcs are examined as long as the product is $\leq 200$
- F: as E, but arcs with an adjacency value less than $\frac{1}{3}$ of the adjacency value of the "best arc so far" are ignored; the arc with the highest adjacency value is examined first

From the results we can see that by using the new arc choice strategies for the Serafini-Ukovich algorithm, some PESP instances could be solved that were impossible to solve by the original algorithm.

| I | Feas. | MIP | SU | | SU† | | SU‡ | | BC |
|---|---|---|---|---|---|---|---|---|---|
| | | time | time | nodes | time | nodes | time | nodes | time |
| 1 | no | 1 s | 1 s | 0 | 1 s | 0 | 1 s | 0 | 1 s |
| 2 | no | 1 s | 1 s | 0 | 1 s | 0 | 1 s | 0 | 1 s |
| 3 | no | 1 s | 1 s | 0 | 1 s | 0 | 1 s | 0 | 1 s |
| 4 | no | 1:34 h | 1:31 h | 12311697 | 0:45 h | 7080808 | 797 s | 488559 | 47 s |
| 5 | yes | 48 s | 183 s | 258255 | 5 s | 9760 | 3 s | 770 | 4 s |
| 6 | ? | ∗ | ∗ | | ∗ | | ∗ | | ∗ |
| 7 | ? | ∗ | ∗ | | ∗ | | ∗ | | ∗ |
| 8 | ? | ∗ | ∗ | | ∗ | | ∗ | | ∗ |
| 9 | ? | ∗ | ∗ | | ∗ | | ∗ | | ∗ |
| 10 | ? | ∗ | ∗ | | ∗ | | ∗ | | ∗ |
| 11 | yes | ∗ | ∗ | | ∗ | | 1783 s | 1262 | 116 s |
| 12 | yes | 1:09 h | 1 s | 1812 | 1 s | 891 | 8 s | 858 | 6 s |
| 13 | ? | ∗ | ∗ | | ∗ | | ∗ | | ∗ |
| 14 | ? | ∗ | ∗ | | ∗ | | ∗ | | ∗ |
| 15 | yes | ∗ | ∗ | | ∗ | | 196 s | 3437 | 13 s |
| 1a | ? | ∗ | ∗ | | ∗ | | ∗ | | ∗ |
| 2a | yes | ∗ | ∗ | | ∗ | | ∗ | | ∗ |
| 3a | yes | ∗ | ∗ | | ∗ | | ∗ | | ∗ |
| 4a | yes | 348 s | ∗ | | 273 s | 583071 | 5 s | 572 | 3 s |
| 5a | yes | ∗ | 1 s | 689 | 1 s | 689 | 8 s | 689 | 7 s |
| 6a | ? | ∗ | ∗ | | ∗ | | ∗ | | ∗ |
| 7a | yes | ∗ | ∗ | | ∗ | | ∗ | | ∗ |
| 8a | ? | ∗ | ∗ | | ∗ | | ∗ | | ∗ |
| 9a | ? | ∗ | ∗ | | ∗ | | ∗ | | ∗ |
| 10a | ? | ∗ | ∗ | | ∗ | | ∗ | | ∗ |
| 11a | yes | ∗ | ∗ | | ∗ | | ∗ | | 300 s |
| 12a | yes | ∗ | ∗ | | 1:11 h | 3600108 | 30 s | 930 | 30 s |
| 13a | ? | ∗ | ∗ | | ∗ | | ∗ | | ∗ |
| 14a | ? | ∗ | ∗ | | ∗ | | ∗ | | ∗ |
| 15a | yes | ∗ | ∗ | | ∗ | | 1121 s | 1440 | 60 s |

Table 5.4: PESP solution times and numbers of search tree nodes (∗: no result after 10 h)

The branch-and-cut method gives even better solution times. With that method, it was possible to find solutions for the instances 3a and 7a within a time limit of one day (which was impossible for the Serafini-Ukovich algorithm or its variants). A solution for instance 2a was found after a few days.

| I | A | | B | | C | | D | | E | | F | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | time | nodes | time | nodes | time | nodes | time | nodes | time | nodes | time | nodes |
| 1 | 1 s | 0 | 1 s | 0 | 1 s | 0 | 1 s | 0 | 1 s | 0 | 1 s | 0 |
| 2 | 1 s | 0 | 1 s | 0 | 1 s | 0 | 1 s | 0 | 1 s | 0 | 1 s | 0 |
| 3 | 1 s | 0 | 1 s | 0 | 1 s | 0 | 1 s | 0 | 1 s | 0 | 1 s | 0 |
| 4 | 1881 s | 500758 | 797 s | 488559 | 0:35 h | 337362 | 0:35 h | 332056 | 0:41 h | 372703 | 1415 s | 212860 |
| 5 | 17 s | 770 | 3 s | 770 | 35 s | 826 | 44 s | 901 | 55 s | 770 | 45 s | 791 |
| 6 | * | | * | | * | | * | | * | | * | |
| 7 | * | | * | | * | | * | | * | | * | |
| 8 | * | | * | | * | | * | | * | | * | |
| 9 | * | | * | | * | | * | | * | | * | |
| 10 | * | | * | | * | | * | | * | | * | |
| 11 | * | | * | | * | | 1783 s | 1262 | 0:53 h | 1150 | 0:49 h | 1264 |
| 12 | 14 s | 860 | 8 s | 858 | 29 s | 858 | 34 s | 858 | 51 s | 858 | 47 s | 927 |
| 13 | * | | * | | * | | * | | * | | * | |
| 14 | * | | * | | * | | * | | * | | * | |
| 15 | 272 s | 1621 | 196 s | 3437 | 902 s | 1460 | 679 s | 1830 | 1359 s | 3099 | 859 s | 1477 |
| 1a | * | | * | | * | | * | | * | | * | |
| 2a | * | | * | | * | | * | | * | | * | |
| 3a | * | | * | | * | | * | | * | | * | |
| 4a | 6 s | 655 | 5 s | 572 | 39 s | 851 | 152 s | 647 | 176 s | 617 | 189 s | 1050 |
| 5a | 12 s | 689 | 8 s | 689 | 61 s | 689 | 197 s | 719 | 362 s | 1082 | 399 s | 718 |
| 6a | * | | * | | * | | * | | * | | * | |
| 7a | * | | * | | * | | * | | * | | * | |
| 8a | * | | * | | * | | * | | * | | * | |
| 9a | * | | * | | * | | * | | * | | * | |
| 10a | * | | * | | * | | * | | * | | * | |
| 11a | * | | * | | * | | * | | * | | * | |
| 12a | 30 s | 930 | 35 s | 930 | 765 s | 10917 | 1020 s | 1026 | 1741 s | 4442 | 1494 s | 1821 |
| 13a | * | | * | | * | | * | | * | | * | |
| 14a | * | | * | | * | | * | | * | | * | |
| 15a | * | | * | | 1121 s | 1440 | 0:41 h | 1437 | 1:20 h | 1517 | 0:54 h | 1404 |

Table 5.5: Results for variants of the Serafini-Ukovich algorithm ($*$: no result after 10 h)

## 5.4 Optimization Results

In this section we present results on the minimum cost scheduling problem of section 2.8 for our real world test instances.

We will at first describe a heuristic method for determining lines and stations where a train change time constraint should be established. For this heuristic, OD-matrices are used. In a second step, we report on experiences with the relaxation iteration algorithm 4.1 and the branch-and-bound algorithm 4.3, assuming that the acceleration methods for the subproblems MCTP and FSP (cf. sections 4.5 and 4.6) are applied. We will then analyze the effects of the acceleration methods in detail. In the last part of this section, we give results for the algorithms 4.4 and 4.5 for the nonlinear problem where the number of used trains depends on the schedule.

### Determining Lines and Stations for Train Change Time Constraints

Let $G = (V, E)$ be the network graph of a railroad network. Let $\omega_{i,j}$ be the number of travelers wishing to travel from station $v_i \in V$ to station $v_j \in V$ in a certain time (e.g. one year; as we have already mentioned, it is very difficult to determine such numbers in practice). The matrix $\Omega$ with entries

$$(\Omega)_{i,j} := \omega_{i,j}$$

is called *origin-destination-matrix* or *OD-matrix*.

A greedy heuristic for determining lines and stations where train change time constraints are useful for many travelers is given by algorithm 5.1. There, a set $\mathcal{X}$ of train change time constraints is constructed. An element of $\mathcal{X}$ consists of a source line $r \in \mathcal{R}$, a destination line $r' \in \mathcal{R}$, a source line direction $\mu \in \{0,1\}$, a destination line direction $\mu' \in \{0,1\}$ and a station $v \in V$ where the train change time has to be provided.

The algorithm requires many MCSP instances to be solved. We have used a variant where the algorithm terminates as soon as $|\mathcal{X}|$ has reached a certain value.

In figure 5.2, the relative amount of travelers in the NS-IR network with a direct connection or with one train change with time constraint (i.e. with a “good connection”) depending on the number of introduced train change time constraints is depicted.

### Direct MIP Solution

We have tried to solve MIP-MCSP instances with the commercial solver CPLEX directly. In table 5.6, the solution time or optimality gap after 10 h computation time is given. An $\infty$-entry means that not even a feasible solution was found in the time limit.

As one can see, only instances with very few train change time constraints can be solved by this method. To be more exact: Our experiments showed that only if the optimal combination of train types and numbers of coaches allowed a feasible solution, the corresponding MIP-MCSP instance could be solved.

**Algorithm 5.1** Determining Lines and Stations for Train Change Time Constraints

$\mathcal{V} := V \times V \backslash \{(v,v) \mid v \in V\}$
$\mathcal{X} := \emptyset$
**loop**
  **if** $\mathcal{V} = \emptyset$ **then**
    Stop. $\mathcal{X}$ has been generated.
  **end if**
  Choose $(v_i, v_j) \in \mathcal{V}$ such that $\omega_{i,j} = \max\{\omega_{k,l} \mid (v_k, v_l) \in \mathcal{V}\}$
  $\mathcal{V} := \mathcal{V} \backslash \{(v_i, v_j)\}$
  **if** there is a line $r \in \mathcal{R}$ connecting $v_i$ and $v_j$ **then**
    **continue**
  **end if**
  **if** traveling from $v_i$ to $v_j$ is possible only with $\geq 2$ train changes **then**
    **continue**
  **end if**
  $\hat{\mathcal{X}} := \{(r_k, \mu_k, r_l, \mu_l, v_m) \mid r_k \in \mathcal{R}, \mu_k \in \{0,1\}, r_l \in \mathcal{R}, \mu_l \in \{0,1\}, v_m \in V$ such that it is possible to travel from $v_i$ to $v_m$ using line $r_k$ in direction $\mu_k$ and to travel from $v_m$ to $v_j$ using line $r_l$ in direction $\mu_l\}$
  **while** $\hat{C} \neq \emptyset$ **do**
    Choose $c \in \hat{\mathcal{X}}$
    $\hat{\mathcal{X}} := \hat{\mathcal{X}} \backslash \{c\}$
    **if** MCSP with train change time constraints from $\mathcal{X} \cup \{c\}$ is solvable without exceeding time limit, iteration limit (for MCSP algorithm 4.1) or node limit (for MCSP algorithm 4.3) **then**
      $\mathcal{X} := \mathcal{X} \cup \{c\}$
      **break**
    **end if**
  **end while**
**end loop**

| Inst. | $\lvert\mathcal{X}\rvert$ | Time (or gap) | Inst. | $\lvert\mathcal{X}\rvert$ | Time (or gap) | Inst. | $\lvert\mathcal{X}\rvert$ | Time (or gap) |
|---|---|---|---|---|---|---|---|---|
| DB-IC | 0 | 1:45 h | DB-IR | 20 | 1549 s | NS-IR | 0 | 1067 s |
| DB-IC | 40 | ∞ | NS-IC | 0 | 1:24 h | NS-IR | 20 | ∞ |
| DB-IR | 0 | 233 s | NS-IC | 40 | ∞ | NS-AR | 0 | 2.7% |

Table 5.6: Results for a direct solution of MIP-MCSP instances

### General Performance of Relaxation Iteration Algorithm 4.1

In table 5.7, results with the relaxation iteration algorithm 4.1 are given. We have tested our instances with several numbers $|\mathcal{X}|$ of train change time constraints. For each combination, the overall computation time, the number of required iterations and the cost (in monetary units) of the optimal schedule (or the best known solution if the time limit was exceeded) are given.

It is assumed that for each iteration of the algorithm, a "small" set $\hat{\mathcal{R}}$ of lines causing the infeasibility

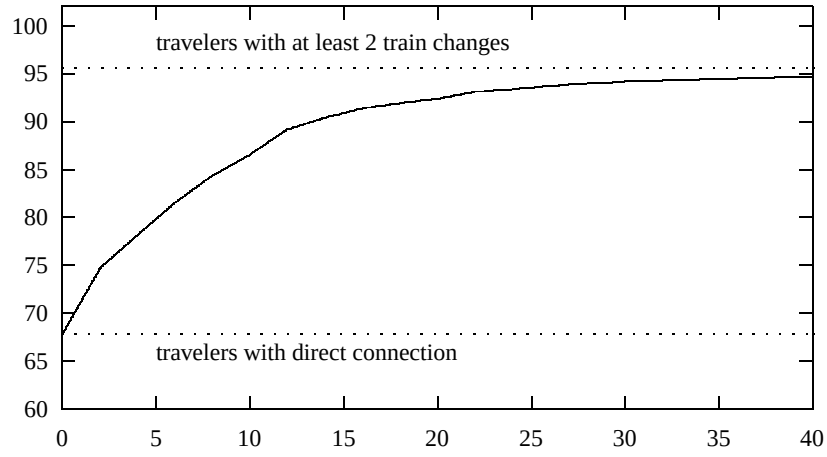



Figure 5.2: Relative amount of travelers with "good connection" (NS-IR)

is generated (cf. chapter 4). If this is not done, many more iterations are required. For example, for the NS-IR instance with $|\mathcal{X}| = 20$, even after 10 h (and over 250 iterations) the lower bound of the second iteration of the algorithm with $\hat{\mathcal{R}}$ was not achieved.

| Inst. | $\lvert\mathcal{X}\rvert$ | Time | # Iter. | Cost | Inst. | $\lvert\mathcal{X}\rvert$ | Time | # Iter. | Cost |
|---|---|---|---|---|---|---|---|---|---|
| DB-IC | 0 | 219 s | 1 | 1.3722 | NS-IC | 30 | 28 s | 1 | 4.0548 |
| DB-IC | 10 | 183 s | 1 | 1.3722 | NS-IC | 40 | $*$ | 235 | $\geq$ 4.0690‡ |
| DB-IC | 20 | 279 s | 1 | 1.3722 | NS-IR | 0 | 33 s | 1 | 2.6984 |
| DB-IC | 30 | 104 s | 1 | 1.3722 | NS-IR | 10 | 27 s | 1 | 2.6984 |
| DB-IC | 40 | $*$ | 925 | $\geq$ 1.3858† | NS-IR | 20 | 989 s | 89 | 2.7792 |
| DB-IR | 0 | 4 s | 1 | 1.7534 | NS-IR | 30 | 1259 s | 91 | 2.7792 |
| DB-IR | 10 | 7 s | 3 | 1.7588 | NS-IR | 40 | 1083 s | 91 | 2.7792 |
| DB-IR | 20 | 11 s | 4 | 1.7592 | NS-AR | 0 | 0:47 h | 1 | 7.4852 |
| DB-IR | 30 | 15 s | 5 | 1.7594 | NS-AR | 10 | 1:04 h | 1 | 7.4852 |
| DB-IR | 40 | 64 s | 22 | 1.7689 | NS-AR | 20 | 1:11 h | 1 | 7.4852 |
| NS-IC | 0 | 30 s | 1 | 4.0548 | NS-AR | 30 | 1:23 h | 1 | 7.4852 |
| NS-IC | 10 | 35 s | 1 | 4.0548 | NS-AR | 40 | 1:24 h | 1 | 7.4852 |
| NS-IC | 20 | 35 s | 1 | 4.0548 | † optimal: 1.3863 ‡ optimal: 4.0709 | | | | |

Table 5.7: Results for the relaxation iteration algorithm 4.1

The increasing solution time per iteration, caused by the additional constraints for the IP-MCTP, is shown in figure 5.3 for the example of the NS-IR network, $|\mathcal{X}| = 20$.

As one can see, the results of the decomposition-based relaxation iteration algorithm leads to much better results. Only two of the test instances could not be solved with this method.

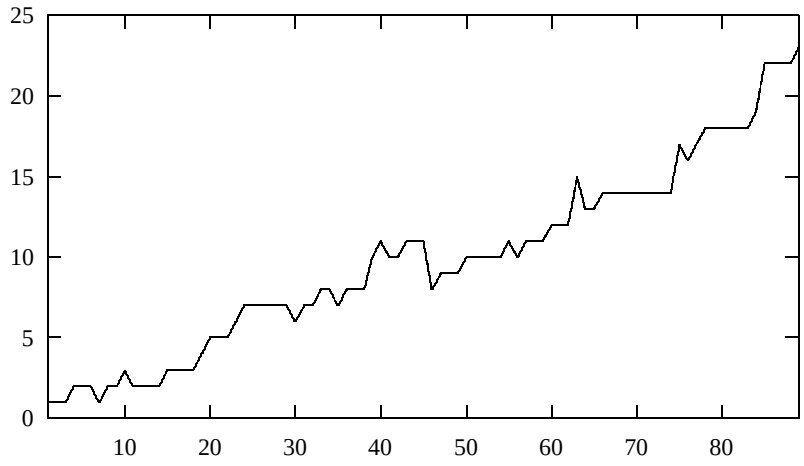



Figure 5.3: Time (in s) required for each iteration (NS-IR, $|\mathcal{X}| = 20$)

### General Performance of Branch-and-bound Algorithm 4.3

The results for the branch-and-bound algorithm 4.3 are shown in table 5.8. There, the solution time, the number of nodes in the search tree or the remaining optimality gap after the time limit and the time required to find the optimal solution (if the problem was solved to optimality within the time limit) are given.

| Inst. | $\lvert\mathcal{X}\rvert$ | Time | # Nodes (or gap) | Time opt. sol. | Inst. | $\lvert\mathcal{X}\rvert$ | Time | # Nodes (or gap) | Time opt. sol. |
|---|---|---|---|---|---|---|---|---|---|
| DB-IC | 0 | 219 s | 1 | 219 s | NS-IC | 30 | 28 s | 1 | 28 s |
| DB-IC | 10 | 183 s | 1 | 183 s | NS-IC | 40 | $*$ | 0.07% | 264 s |
| DB-IC | 20 | 279 s | 1 | 279 s | NS-IR | 0 | 33 s | 1 | 33 s |
| DB-IC | 30 | 104 s | 1 | 104 s | NS-IR | 10 | 27 s | 1 | 27 s |
| DB-IC | 40 | 9:31 h | 37746 | 480 s | NS-IR | 20 | 1:00 h | 1486 | 110 s |
| DB-IR | 0 | 4 s | 1 | 4 s | NS-IR | 30 | 1:19 h | 2017 | 135 s |
| DB-IR | 10 | 14 s | 5 | 14 s | NS-IR | 40 | 1:20 h | 2009 | 132 s |
| DB-IR | 20 | 15 s | 8 | 8 s | NS-AR | 0 | 0:47 h | 1 | 0:47 h |
| DB-IR | 30 | 27 s | 18 | 12 s | NS-AR | 10 | 1:04 h | 1 | 1:04 h |
| DB-IR | 40 | 122 s | 103 | 51 s | NS-AR | 20 | 1:11 h | 1 | 1:11 h |
| NS-IC | 0 | 30 s | 1 | 30 s | NS-AR | 30 | 1:23 h | 1 | 1:23 h |
| NS-IC | 10 | 35 s | 1 | 35 s | NS-AR | 40 | 1:24 h | 1 | 1:24 h |
| NS-IC | 20 | 35 s | 1 | 35 s | | | | | |

Table 5.8: Results for the branch-and-bound algorithm 4.3

In order to get an idea of the effect of the acceleration methods leading from the simple branch-and-bound algorithm 4.2 to algorithm 4.3, the solution times and number of nodes for some instances for the simple algorithm 4.2 are shown in table 5.9.

| Inst. | $\|\mathcal{X}\|$ | Time | Nodes (gap) | Inst. | $\|\mathcal{X}\|$ | Time | Nodes (gap) | Inst. | $\|\mathcal{X}\|$ | Time | Nodes (gap) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| DB-IC | 40 | $*$ | 0.01% | DB-IR | 40 | 132 s | 132 | NS-IR | 20 | 1:20 h | 2105 |

Table 5.9: Results for the simple branch-and-bound algorithm 4.4

The branch-and-bound algorithm gives feasible solutions in a few seconds or minutes for our test instances. Moreover, the quality of these solutions is quite acceptable (after a few minutes, the optimality gap was less than 1% in our test cases). However, the algorithm is slower than the relaxation iteration algorithm (if it does not terminate with an optimal solution at the root node).

### Solving MCTP instances

We will now discuss the different methods for solving MCTP instances. As example instances, we have chosen the MCTP instances from the first iteration of algorithm 4.1 (or the root node from algorithm 4.3 respectively).

In table 5.10, the number of rows, columns and non-zeros of the MIPs, the relative gap between the optimal LP solution and the optimal MIP solution and the solution time for the general integer model IP-MCTP and the binary model BP-MCTP are given.

The solver CPLEX contains a MIP preprocessor (see [34]) for reducing the MIP size. The numbers from table 5.10 were obtained after using the preprocessor. We have also tested several variable branching strategies (cf. appendix B). The solution times are always given for the fastest strategy for the particular instance.

| Inst. | Integer model IP-MCSP | | | | | Inst. | Binary model BP-MCSP | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | #Con. | #Var. | $\#\neq 0$ | Root gap | Time | | #Con. | #Var. | $\#\neq 0$ | Root gap | Time |
| DB-IC | 204 | 244 | 1304 | 2.4% | 4 s | DB-IC | 74 | 735 | 3435 | 0.9% | 1 s |
| DB-IR | 319 | 413 | 1436 | 2.4% | 1 s | DB-IR | 69 | 471 | 1376 | 0.3% | 1 s |
| NS-IC | 148 | 179 | 826 | 2.5% | 129 s | NS-IC | 57 | 793 | 3237 | 2.5% | 20 s |
| NS-IR | 86 | 107 | 570 | 1.5% | 5 s | NS-IR | 41 | 545 | 2741 | 1.5% | 2 s |
| NS-AR | 471 | 617 | 2618 | 2.8% | † | NS-AR | 178 | 2221 | 9328 | 2.1% | 865 s |

†: terminates with memory failure

Table 5.10: Results for different MCTP models

The effect of using the cutting planes from section 4.5 can be seen in table 5.11. There, the number of cutting plane iterations before starting the MIP branch-and-bound process, the number of used cuts, the relative gap between the LP solution and the MIP solution and the solution time are given.

| I | # Iter. | # Cuts | Root gap | Time | I | # Iter. | # Cuts | Root gap | Time |
|---|---|---|---|---|---|---|---|---|---|
| DB-IC | 3 | 7 | 0.6% | 1 s | DB-IR | 2 | 18 | 0.05% | 1 s |
| NS-IC | 10 | 134 | 0.6% | 9 s | NS-IR | 10 | 94 | 0.9% | 4 s |
| NS-AR | 9 | 123 | 1.1% | 934 s | | | | | |

Table 5.11: Results for the BP-MCTP with cutting plane algorithm

We can see that for our instances, the BP-MCTP formulation gives better results (i.e. solution times) than the IP-MCTP formulation. The use of cutting planes provides an additional acceleration in some cases.

### Solving FSP instances

In order to compare the different FSP algorithms resulting from the variants of the algorithm of Serafini and Ukovich, we have tested them by integrating them into the relaxation iteration algorithm 4.1 and the branch-and-bound algorithm 4.3. Our MCSP instances lead to the FSP instance sizes from table 5.12. Note that for all instances arising from the same MCSP instances, these numbers are identical.

| Inst. | $\lvert\mathcal{X}\rvert$ | $\lvert V_{\mathcal{G}}\rvert$ | $\lvert A_{\mathcal{G}}\rvert$ | $\lvert\mathcal{J}\rvert$ | Inst. | $\lvert\mathcal{X}\rvert$ | $\lvert V_{\mathcal{G}}\rvert$ | $\lvert A_{\mathcal{G}}\rvert$ | $\lvert\mathcal{J}\rvert$ |
|---|---|---|---|---|---|---|---|---|---|
| DB-IC | 40 | 924 | 1529 | 298 | NS-IR | 20 | 488 | 941 | 224 |
| DB-IR | 20 | 2112 | 2295 | 116 | NS-AR | 0 | 1968 | 3956 | 1012 |
| NS-IC | 40 | 496 | 783 | 134 | | | | | |

Table 5.12: FSP instance sizes

We will now discuss the results for different variants of FSP algorithms derived from variants of the Serafini-Ukovich algorithm. We have applied these variants to the first 200 nodes of the branch-and-bound tree of algorithm 4.3 for our test instances (if there were so many nodes).

Table 5.13 shows detailed results for the different algorithms. We have separated feasible and infeasible instances and given minimal, maximal, average solution time and the medians of the solution times. Our algorithmic variants are:

- SU: original algorithm of Serafini and Ukovich
- SU†: Serafini and Ukovich algorithm with arc preorder by number of feasible modulo parameters
- SU‡: Serafini and Ukovich algorithm with arc choice strategy A from table 5.5
- SU*: Serafini and Ukovich algorithm with arc choice strategy B from table 5.5

| MCSP Inst. | $\|\mathcal{X}\|$ | Algo. | Feasible Instances | | | | | Infeasible Instances | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | # FSP Inst. | Solution Time | | | | # FSP Inst. | Solution Time | | | |
| | | | | Min. | Max. | Avg. | Med. | | Min. | Max. | Avg. | Med. |
| DB-IC | 40 | SU | 1 | ∗ | ∗ | ∗ | ∗ | 1 | 1 s | 1 s | 1 s | 1 s |
| DB-IC | 40 | SU† | 0 | | | | | 4 | 1 s | ∗ | ∗ | 1 s |
| DB-IC | 40 | SU‡ | 1 | 253 s | 253 s | 253 s | 253 s | 40 | 1 s | 14 s | 5 s | 5 s |
| DB-IC | 40 | SU* | 1 | ∗ | ∗ | ∗ | ∗ | 13 | 1 s | 3 s | 2 s | 2 s |
| DB-IR | 10 | SU | 2 | 2 s | 2 s | 2 s | 2 s | 2 | 1 s | 1 s | 1 s | 1 s |
| DB-IR | 10 | SU† | 2 | 2 s | 2 s | 2 s | 2 s | 2 | 1 s | 1 s | 1 s | 1 s |
| DB-IR | 10 | SU‡ | 2 | 4 s | 5 s | 4 s | 4 s | 2 | 1 s | 1 s | 1 s | 1 s |
| DB-IR | 10 | SU* | 2 | 4 s | 4 s | 4 s | 4 s | 2 | 1 s | 1 s | 1 s | 1 s |
| NS-IC | 40 | SU | 1 | 1 s | 1 s | 1 s | 1 s | 55 | 1 s | 1 s | 1 s | 1 s |
| NS-IC | 40 | SU† | 1 | 9 s | 9 s | 9 s | 9 s | 75 | 1 s | 42 s | 8 s | 1 s |
| NS-IC | 40 | SU‡ | 1 | 7 s | 7 s | 7 s | 7 s | 77 | 1 s | 5 s | 2 s | 1 s |
| NS-IC | 40 | SU* | 1 | 2 s | 2 s | 2 s | 2 s | 78 | 1 s | 2 s | 1 s | 1 s |
| NS-IR | 20 | SU | 0 | | | | | 200 | 1 s | 1 s | 1 s | 1 s |
| NS-IR | 20 | SU† | 1 | 1 s | 1 s | 1 s | 1 s | 93 | 1 s | 602 s | 3 s | 1 s |
| NS-IR | 20 | SU‡ | 1 | 48 s | 48 s | 48 s | 48 s | 95 | 1 s | 3 s | 1 s | 1 s |
| NS-IR | 20 | SU* | 1 | 23 s | 23 s | 23 s | 23 s | 95 | 1 s | 3 s | 1 s | 1 s |
| NS-AR | 0 | SU | 1 | ∗ | ∗ | ∗ | ∗ | 0 | | | | |
| NS-AR | 0 | SU† | 1 | 92 s | 92 s | 92 s | 92 s | 0 | | | | |
| NS-AR | 0 | SU‡ | 1 | 1:10 h | 1:10 h | 1:10 h | 1:10 h | 0 | | | | |
| NS-AR | 0 | SU* | 1 | 0:34 h | 0:34 h | 0:34 h | 0:34 h | 0 | | | | |

Table 5.13: Results for FSP instances with different algorithms

## Algorithms for the Nonlinear Problem

We have also tried to solve the N-MCSP with methods like algorithm 4.4 and 4.5. As we have already mentioned in section 4.7, these algorithms are very slow. Results with our implementation of a simplified version of these algorithms are shown in table 5.14 and table 5.15.

There are many instances for which even the first FSP instance with optimization could not be solved (the first BP-MCTP instance was always solved in a few seconds or a few minutes).

As one can see from the tables, the exact calculation reveals that the numbers of trains are overestimated to a certain extent by using $\hat{t}_{r,\tau}$ instead of $t_{r,\tau}$.

| Inst. | $\|\mathcal{X}\|$ | Time | # Iter. | Cost or gap | Inst. | $\|\mathcal{X}\|$ | Time | # Iter. | Cost or gap |
|---|---|---|---|---|---|---|---|---|---|
| DB-IC | 0 | 1161 s | 2 | 1.3396 | NS-IC | 30 | $*$ | 17 | 0.3% |
| DB-IC | 10 | $*$ | 1 | 3.6% | NS-IC | 40 | $*$ | 270 | $\geq$ 3.9268 |
| DB-IC | 20 | $*$ | 18 | 1.3% | NS-IR | 0 | 0:55 h | 2 | 2.5116 |
| DB-IC | 30 | $*$ | 11 | 0.4% | NS-IR | 10 | 1:00 h | 2 | 2.5116 |
| DB-IC | 40 | $*$ | 896 | $\geq$ 1.3544 | NS-IR | 20 | $*$ | 307 | 0.4% |
| DB-IR | 0 | 277 s | 2 | 1.6968 | NS-IR | 30 | $*$ | 507 | $\geq$ 2.6190 |
| DB-IR | 10 | $*$ | 124 | 0.04% | NS-IR | 40 | $*$ | 499 | $\geq$ 2.6192 |
| DB-IR | 20 | $*$ | 630 | 0.04% | NS-AR | 0 | $*$ | 1 | $\geq$ 6.7253 |
| DB-IR | 30 | $*$ | 27 | 0.04% | NS-AR | 10 | $*$ | 1 | $\geq$ 6.7253 |
| DB-IR | 40 | $*$ | 166 | 0.2% | NS-AR | 20 | $*$ | 1 | $\geq$ 6.7253 |
| NS-IC | 0 | 928 s | 2 | 3.9075 | NS-AR | 30 | $*$ | 1 | $\geq$ 6.7253 |
| NS-IC | 10 | $*$ | 1 | 0.6% | NS-AR | 40 | $*$ | 1 | $\geq$ 6.7253 |
| NS-IC | 20 | 1084 s | 2 | 3.9075 | | | | | |

Table 5.14: Results for the relaxation iteration algorithm 4.4 for the nonlinear problem

| Inst. | $\|\mathcal{X}\|$ | Time | # Nodes | Cost (or gap) | Inst. | $\|\mathcal{X}\|$ | Time | # Nodes | Cost (or gap) |
|---|---|---|---|---|---|---|---|---|---|
| DB-IC | 0 | 1161 | 2 | 1.3396 | NS-IC | 30 | $*$ | 4 | 0.8% |
| DB-IC | 10 | $*$ | 1 | 3.6% | NS-IC | 40 | $*$ | 4065 | 0.7% |
| DB-IC | 20 | $*$ | 6 | 0.1% | NS-IR | 0 | 0:55 h | 1 | 2.5116 |
| DB-IC | 30 | $*$ | 6 | 0.4% | NS-IR | 10 | 1:00 h | 1 | 2.5116 |
| DB-IC | 40 | $*$ | 3680 | $\geq$ 1.3544 | NS-IR | 20 | $*$ | 513 | 1.1% |
| DB-IR | 0 | 277 s | 1 | 1.6968 | NS-IR | 30 | $*$ | 7134 | 1.4% |
| DB-IR | 10 | $*$ | 124 | 0.04% | NS-IR | 40 | $*$ | 4785 | $\geq$ 2.6174 |
| DB-IR | 20 | $*$ | 51 | 0.02% | NS-AR | 0 | $*$ | 1 | $\geq$ 6.7253 |
| DB-IR | 30 | $*$ | 51 | 0.02% | NS-AR | 10 | $*$ | 1 | $\geq$ 6.7253 |
| DB-IR | 40 | $*$ | 430 | 0.2% | NS-AR | 20 | $*$ | 1 | $\geq$ 6.7253 |
| NS-IC | 0 | 928 s | 1 | 3.9075 | NS-AR | 30 | $*$ | 1 | $\geq$ 6.7253 |
| NS-IC | 10 | $*$ | 1 | 0.6% | NS-AR | 40 | $*$ | 1 | $\geq$ 6.7253 |
| NS-IC | 20 | 1084 s | 1 | 3.9075 | | | | | |

Table 5.15: Results for the branch-and-bound algorithm 4.5 for the nonlinear problem

# Chapter 6

# Conclusions and Suggestions for Further Research

In this thesis, we have presented and developed models and algorithms for generating and optimizing train schedules. From a *theoretical* point of view, these problems belong to a class of *very hard* problems.
Despite this fact, for *practical* instances, our algorithms perform quite satisfactory, i.e. we can find *optimal* solutions for small or medium sized networks like the InterCity or InterRegio networks of Germany or the Netherlands. Our relaxation-based algorithms produce solutions with *provable quality* in a few minutes. It is worth mentioning that *theoretical* considerations on the problem structure were of great help when designing *practical* algorithms.
For larger networks, a *decomposition* into *regional networks* seems to be adequate. For these subnetworks, solutions can be produced that have to be *combined* to an overall solution. This will not be possible directly in some cases, but require small *adaptations* that have to be performed *manually*. This is also the traditional way of generating schedules.
The same holds for another obvious goal, namely the *combination* of *supply networks.* Since lines from different supply networks often use the same physical railroad tracks, their schedules cannot be considered separately. Again, one can try to solve the combined problem directly or by a decomposition method.

A practical requirement that has to be taken into account in the future is the consideration of *multiple objectives.* We have seen several evaluation criteria of practical relevance for schedules, including *minimization of travel time*, *maximization of robustness* or *minimization of cost.*
Our model considers the *minimization of operational cost* directly. Aspects like minimization of total travel time are only taken into consideration *indirectly* by setting an upper bound on the time for train changes. The fact that long waiting times at stations may require an additional (costly) train composition also leads to an indirect reduction of travel time.
There are several principle approaches for considering multiple objectives simultaneously:

- *Weighted sum of objectives*: We can construct combined models with an objective function being the weighted sum of the original objective functions. In practice, this approach often leads to unsatisfactory results. That objective with the highest weight is considered only, regardless of the other objectives.

- *Constraints for some criteria*: Another common method is the optimization of only one criterion subject to other criteria requiring a certain "acceptance level". In this case, constraints are given for those objectives that are ignored by the optimization.

  The MCSP can be interpreted as such an approach. The operational costs are minimized while the train change time must not exceed a certain limit. A similar approach is the following.

- *Pareto optimal solutions*: A feasible solution of a multi-objective problem is called *pareto optimal* if all other feasible solution with a better objective value for a single criterion have worse objective values for at least one other criterion.

  Our experiments with several sets of train change time constraints $\mathcal{X}$ show pareto optimal solutions concerning the objectives *travelers with "good" connection* and *cost*, see figure 6.1 (this is not absolutely correct because we have used a heuristic to determine the set $\mathcal{X}$).

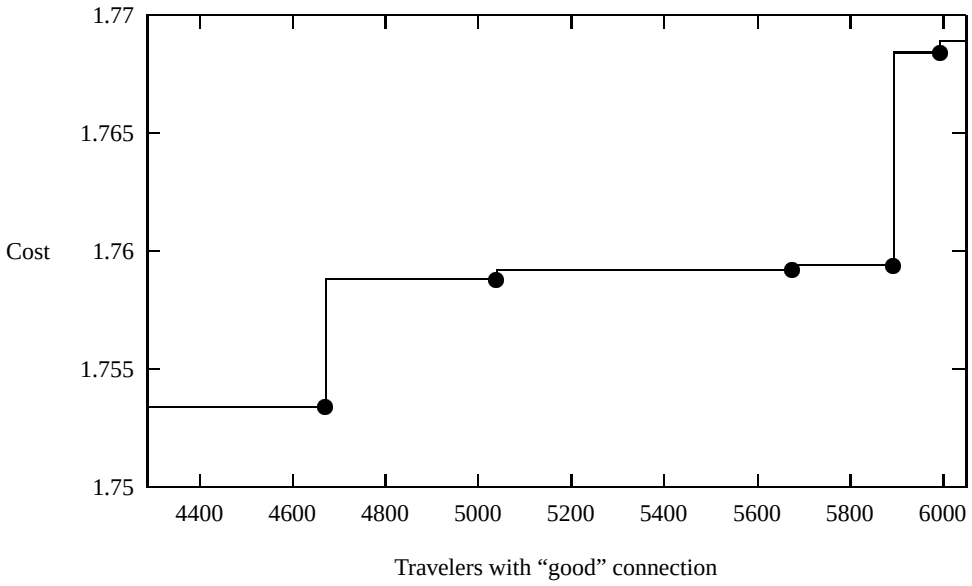



Figure 6.1: Pareto optimal solutions for the German InterRegio network

Another problem we have already mentioned is that a *change* in the transport *service*, like another line plan or schedule, affects the *travelers' behavior*. The passenger demand data used for the generation of a schedule may actually become worthless by the introduction of that schedule. In order to overcome this problem, practitioners try to *simulate* the travelers' behavior and thus try to estimate the actual effect of a schedule (or line plan etc.).

Often, an *iterative* approach is used: After a schedule is obtained, the demand data is updated according to a simulation. Afterwards another schedule is generated. One may hope that this method *converges*, although there is, of course, no *mathematical justification* for such a behavior.

Naturally, the final goal is transport planning *without* the *hierarchical decomposition* from figure 1.2 on page 2. However, this goal seems to be out of reach at the moment, due to each step being still a hard problem for real-world-sized instances.

# Appendix A

# Computational Complexity

A topic of main interest for designing algorithms is the question "How long does it take in the worst case to solve a problem instance of a certain size?". Another point may be the amount of memory that is needed. An approach to answer such questions is given by the theory of computational complexity of algorithms. There is a lot of literature concerning this subject, for example [1, 25, 51]. We will give a short description of some ideas on computational complexity here.

The *size* of a problem instance can be understood as the number of bits that is needed to describe all the data that defines the instance. For example, the size of an integer $i \neq 0$ is $\text{size}(i) = 1 + \lceil \log_2 |i| \rceil$ in this case. The *running time* of an algorithm may be measured as the number of "basic steps" like assignment steps, addition, subtraction, multiplication, division, comparison of two numbers, that are required to solve a problem instance. In general, the running time depends on the size of the instance.

In order to define the worst case complexity of an algorithm, the "big O" notation is used. Let $f : \mathbb{N} \to \mathbb{R}$ be a function. An algorithm is said to have worst case running time (or *complexity*) of $O(f(n))$ if there are constants $c > 0$ and $n_0 \in \mathbb{N}$ such that the running time does not exceed $c \cdot f(n)$ for each instance of size $n \geq n_0$.

## A.1 The Problem Classes P and NP

An algorithm is called *polynomial time algorithm* if there exists a polynomial $f$ such that the algorithm has a running time of $O(f(n))$. An algorithm is called *exponential time algorithm* if its running time can only be bounded by an exponential function, but not by a polynomial.

We will focus only on *decision problems* in the following. A problem is called decision problem if the answer consists only of an answer "yes" or "no". A minimization problem can be solved by using decision problem algorithms via binary search techniques ("is there a solution with an objective value $\leq c$"). Therefore, if there is a polynomial time algorithm to solve a decision problem, then there is also a polynomial time algorithm for such a corresponding optimization problem.

The set of all decision problems for which a polynomial time solution algorithm exists, is denoted by P.

Many important decision problems (and therefore many important optimization problems) are not known to have polynomial time solution algorithms. However, for many problems, a situation like

in the following example is given: Consider a mixed integer programming problem (cf. appendix B) and the question if there is a solution $\boldsymbol{x}$ with an objective value $\leq c$. If we are given a solution $\boldsymbol{x}$ with objective value $\leq c$, we can check that the answer to our problem is "yes" in polynomial time (simply by evaluating the objective function for $\boldsymbol{x}$). Note that on the other hand, if we are given a solution $\boldsymbol{x}$ with objective value $> c$, we cannot verify that the answer is "no" that easily. This motivates the following definition of the problem class NP.

A decision problem is said to be in the class NP if and only if, for every instance for which the answer is "yes", there is a *certificate*, namely a binary string whose length is polynomially bounded by the size of the input data, and a polynomial time algorithm which, when supplied with the input data of the problem instance and the certificate, confirms that the answer is indeed "yes" in polynomial time.

In our example, this certificate would consist of the binary encoding of $\boldsymbol{x}$. Note that, if the answer for an instance is "no", there is nothing said about certificates or polynomial time algorithms.

## A.2 NP-complete Problems

Consider two problems $\Pi$ and $\Pi'$. $\Pi$ is said to be *polynomially transformable* to $\Pi'$, if there is an algorithm which, for every instance $I$ of $\Pi$, constructs an instance $I'$ of $\Pi'$ (i.e. it takes the input data from $I$ and constructs the input data for $I'$) in polynomial time such that the answer to $I'$ is "yes" if and only the answer for $I$ is "yes".

A problem is said to be NP-*complete* if and only if it is in NP and every problem in NP can be polynomially transformed to it.

Many important decision problems for which no polynomial time solution algorithm is known (for example the decision version of solving mixed integer programs) have been shown to be NP-complete. In [25], there is a list of problems which were known to be NP-complete already in 1979.

If there is a polynomial time solution algorithm for a single NP-complete problem, then there are polynomial time solution algorithms for all NP-complete problems. In this case, we would have $\mathrm{P} = \mathrm{NP}$, which is hardly believed. Therefore, if we can show that a problem is NP-complete, we have reason to believe that there is no polynomial time for solving it.

In order to show that a problem $\Pi$ is NP-complete it is sufficient to show that it is in NP and that there is an NP-complete problem that can be polynomially transformed to $\Pi$.

# Appendix B

# Mixed Integer Linear Programs

## B.1 Linear and Mixed Integer Linear Programs

The problem of the form

$$\begin{array}{ll} \text{given} & A \in \mathbb{R}^{m\times n},\ \boldsymbol{b} \in \mathbb{R}^m,\ \boldsymbol{c} \in \mathbb{R}^n \\ \text{minimize} & \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to} & A\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \in \mathbb{R}^n \end{array} \tag{B.1}$$

is called linear programming problem or, for short, *linear program* (LP). The set $\{\boldsymbol{x} \in \mathbb{R}^n \mid A\boldsymbol{x} \leq \boldsymbol{b}\}$ is called *feasible set* or *feasible region.* The function $f : \mathbb{R}^n \to \mathbb{R}$ defined by $f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x}$ is called *objective function.* For the theory of linear programs and solution methods, we refer to [15] and [55].

If some components of $x$ are requested to have integer values, the problem is called *mixed integer linear program* (MIP). If all components have to be integer, it is an *integer linear program* (IP). The corresponding feasible sets are given by the intersection of $\{\boldsymbol{x} \mid A\boldsymbol{x} \leq \boldsymbol{b}\}$ and the integrality constraints. Some literature that deals with such problems is [46, 55].

## B.2 Polyhedra

In this section some well known results and notions of *polyhedral theory* are presented. A more comprehensive discussion can be found in [46].

A *polyhedron* $P \subseteq \mathbb{R}^n$ is the set of points satisfying a finite number of linear inequalities, i.e. a set that can be described as $\{\boldsymbol{x} \in \mathbb{R}^n \mid A\boldsymbol{x} \leq \boldsymbol{b},\ A \in \mathbb{R}^{m\times n},\ \boldsymbol{b} \in \mathbb{R}^m\}$. If a polyhedron $P$ is bounded (that is, if there is an $\omega \in \mathbb{R}$ such that $P \subseteq \{\boldsymbol{x} \in \mathbb{R}^n \mid -\omega \leq x_i \leq \omega$ for each $i \in \{1,\ldots,n\}\}$ where $x_i$ are the components of $\boldsymbol{x}$), it is also called *polytope.* A polyhedron $P$ is of *dimension* $k$, denoted by $\dim P = k$, if the maximum number of affinely independent points in $P$ is $k+1$.

An inequality $\boldsymbol{\alpha}^T\boldsymbol{x} \leq \alpha_0$, $\boldsymbol{\alpha} \in \mathbb{R}^n$, $\alpha_0 \in \mathbb{R}$ is called *valid* inequality for $P$ if $P \subseteq \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{\alpha}^T\boldsymbol{x} \leq \alpha_0$. If $\boldsymbol{\alpha}^T\boldsymbol{x} \leq \alpha_0$ is a valid inequality for $P$, then $F = \{\boldsymbol{x} \in P \mid \boldsymbol{\alpha}^T\boldsymbol{x} = \alpha_0\}$ is called a *face* of $P$. In this case $\boldsymbol{\alpha}^T\boldsymbol{x} \leq \alpha_0$ is said to *generate* $F$.

A face $F$ is said to be *proper* if $F \neq \emptyset$ and $F \neq P$. A face $F$ is called *facet* if $\dim F = \dim P - 1$. The single point of a zero dimensional face $F = \{\boldsymbol{x}_*\}$ of a polytope $P$ is called *extreme point* of $P$. A point $\boldsymbol{x}_* \in P$ is an extreme point of a polytope $P$ if and only if there do not exist $\boldsymbol{x}', \boldsymbol{x}'' \in P$ such that $\boldsymbol{x} = \frac{1}{2}\boldsymbol{x}' + \frac{1}{2}\boldsymbol{x}''$.

The feasible set of LP is a polyhedron. In general, the feasible set of MIP or IP is *not* a polyhedron. Assume now that the feasible set is a polytope. Many MIP solution algorithms start by solving the corresponding LP (the so called *LP relaxation*, cf. section B.3) and finding an optimal extreme point (note that if there is an optimal point in a polytope, then there is also an optimal extreme point). If this point does not satisfy the integrality constraints, which is the normal case, the algorithms start some other, usually time consuming procedure.

Let $P = \{\boldsymbol{x} \in \mathbb{R}^n \mid A\boldsymbol{x} \leq \boldsymbol{b}\}$ be a polytope. From Weyl's theorem, we know that the convex hull of the feasible set $\operatorname{conv}(\{\boldsymbol{x} \in \mathbb{Z}^n \mid A\boldsymbol{x} \leq \boldsymbol{b}\}) =: C$ for the corresponding IP can be described as a polytope $\{\boldsymbol{x} \in \mathbb{R}^n \mid A'\boldsymbol{x} \leq \boldsymbol{b}'\}$ with $A' \in \mathbb{R}^{m' \times n}$, $\boldsymbol{b}' \in \mathbb{R}^{m'}$. The extreme points of this convex hull all satisfy the integrality constraints. If $A'$ and $\boldsymbol{b}'$ were known, one could start the solution algorithm with $A'$ and $b'$, and it would only need to solve the LP relaxation to give an optimal solution for IP, cf. [46]. A similar statement can be given for MIP.

Unfortunately it is very difficult to find $A'$ and $\boldsymbol{b}'$ for a given set $\{\boldsymbol{x} \in \mathbb{Z}^n \mid A\boldsymbol{x} \leq \boldsymbol{b}\}$ if only $A$ and $\boldsymbol{b}$ are known, which is the usual case. According to [46], for each facet of the polytope $C$, a valid inequality is necessary in a description $\{\boldsymbol{x} \in \mathbb{R}^n \mid A'x \leq \boldsymbol{b}'\}$ for $C$, and there is no polynomial $\phi$ such that the number of facets of $C$ is bounded by $\phi(\operatorname{size}(A, \boldsymbol{b}))$, cf. [55].

Therefore, it is practically impossible to find all facets of $C$. However, one can try to find some facets or at least valid inequalities for $C$ heuristically. Consider the example of figure B.1. In the middle, the feasible region $P$ for the LP relaxation is given. On the right, one can see the convex hull $C$ of the feasible set of IP. After adding the constraint $2x_1 + x_2 \leq 3$ (i.e. the dashed constraint), the optimal solution of LP is integral although this inequality is not even a facet.

$$
\begin{array}{lrrrrl}
\min & & & - & x_2 & \\
\text{s.t.} & - & \frac{1}{2}x_1 & + & x_2 & \leq \frac{1}{2} \\
& - & \frac{1}{4}x_1 & - & x_2 & \leq \frac{1}{4} \\
& & 4x_1 & - & x_2 & \leq 6 \\
& & x_1, & & x_2 & \in \mathbb{Z}
\end{array}
$$

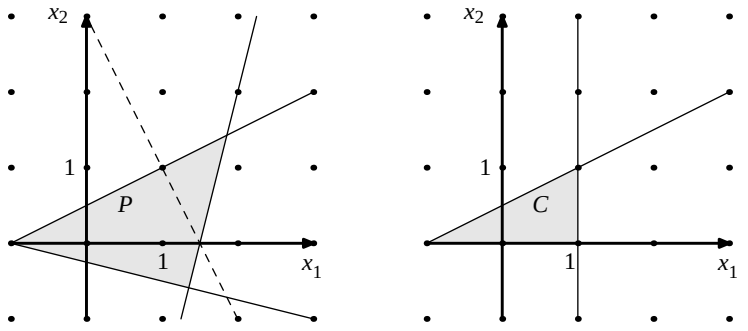



Figure B.1: IP, feasible region without integrality constraints, convex hull of feasible region

In a so called *cutting plane algorithm* (cf. section B.3) for IP solution, the first step consists of solving the LP relaxation of the problem instance. Let the optimal point for the LP relaxation be $\boldsymbol{x}_*$. If it is integral, the IP solution has been found. Otherwise one tries to find a valid inequality $\boldsymbol{\alpha}^T\boldsymbol{x} \leq \alpha_0$ for the convex hull of the feasible set such that $\boldsymbol{\alpha}^T\boldsymbol{x}_* > \alpha_0$. This inequality is added to the LP relaxation and the new linear problem is solved. The process is continued iteratively, until an integer solution

has been found (see section B.3).

For this type of algorithm it is important that, given a set of points $S$ and an additional point $\boldsymbol{x}_*$, one can decide whether $\boldsymbol{x}_* \in \operatorname{conv} S$ and in case of $\boldsymbol{x}_* \notin \operatorname{conv} S$ give a valid inequality for $\operatorname{conv} S$ which is violated by $\boldsymbol{x}$. This problem is called *separation problem* and can in general be polynomially transformed into the original optimization problem (and vice versa), see [29]. Nevertheless, in many special cases some classes of facets or valid inequalities can be found in polynomial time.

## B.3 Solution Methods

In the following, we assume having an IP instance $\min\{\boldsymbol{c}^T\boldsymbol{x} \mid A\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \in \mathbb{Z}^n\}$. The presented methods can be easily extended for the solution of MIP instances. Let $S := \{\boldsymbol{x} \in \mathbb{Z}^n \mid A\boldsymbol{x} \leq \boldsymbol{b}\}$ (i.e. $S$ is the feasible set for the instance).

### Relaxation Iteration Algorithms

One class of solution methods are *relaxation iteration algorithms*, see algorithm B.1. These algorithms try to minimize the objective function on a set $R \supseteq S$. If an optimal solution $\boldsymbol{x}_*$ is found with $\boldsymbol{x}_* \in S$, then $\boldsymbol{x}_*$ is an optimal solution for the IP instance. Otherwise $R$ is replaced by a set $R'$ with $R \supsetneq R' \supseteq S$ and the procedure is restarted. In most cases these algorithms are designed in such a way that during the solution of $\min\{\boldsymbol{c}^T\boldsymbol{x} \mid \boldsymbol{x} \in R\}$, information generated by previous iterations can be reused.

**Algorithm B.1** Relaxation Iteration Algorithm

```
Choose R ⊇ S.
loop
  if R = ∅ then
    Stop. The problem is infeasible.
  end if
  Calculate z* = min{c^T x | x ∈ R} and a corresponding solution x*.
  if x* ∈ S then
    Stop. x* is an optimal solution.
  end if
  Choose R' with R ⊋ R' ⊇ S.
  R := R'.
end loop
```

An important example for this type of algorithms are *fractional cutting plane algorithms* (or, for short, *cutting plane algorithms*). Here, $R = \{\boldsymbol{x} \in \mathbb{R}^n \mid A\boldsymbol{x} \leq \boldsymbol{b}\}$ is chosen initially, i.e. the corresponding LP instance (the so called *LP relaxation*) is solved. If $\boldsymbol{x}_* \notin S$, which means that $\boldsymbol{x}$ has a fractional component, $R'$ is chosen as the intersection of $R$ with an additional linear inequality (the *cutting plane*) which is violated by $\boldsymbol{x}_*$, but valid for every $\boldsymbol{x} \in S$. One can show that there is always such an inequality and that these inequalities can be chosen in such a way that the algorithm terminates after a finite number of iterations, see [28] or [46].

There are cutting plane algorithms working with a particular set of cutting planes such that there is not always an inequality violated by $\boldsymbol{x}_*$, but valid for each $\boldsymbol{x} \in S$. They proceed with other techniques as soon as in some iteration, there is no cutting plane violated by $\boldsymbol{x}_*$, but valid for each $\boldsymbol{x} \in S$.

One motivation for using cutting plane algorithms is the fact that after adding an inequality to $R$, the solution $\boldsymbol{x}_*$ is still dually feasible, and the minimization in the next iteration can be done from an advanced basis for the dual simplex algorithm (cf. [46]). Another advantage is that in every iteration, $z_*$ is a lower bound on the optimal solution of the original IP instance.

### Enumerative Algorithms

Another class of algorithms often used for the solution of MIPs is given by *enumerative algorithms.* Let $S = \bigcup_{\rho=1}^{r} S_r$ with $S_{\rho_1} \cap S_{\rho_2} = \emptyset$ if $\rho_1 \neq \rho_2$, $\rho_1 \in \{1,\ldots,r\}$, $\rho_2 \in \{1,\ldots,r\}$. Then $\{S_\rho \mid \rho = 1,\ldots,r\}$ is said to be a *partition* of $S$. The fact that

$$\min\{\boldsymbol{c}^T\boldsymbol{x} \mid \boldsymbol{x} \in S\} = \min_{\rho\in\{1,\ldots,r\}} \{\boldsymbol{c}^T\boldsymbol{x} \mid \boldsymbol{x} \in S_\rho\}$$

suggests using a divide-and-conquer algorithm. The set $S_\rho$ may be partitioned again, and a partition tree structure is obtained. The way of partitioning $S$ is of course the crucial point for the algorithm. If $S$ was partitioned into sets that contain only one element (the extreme case), we would have a complete enumeration. This procedure usually exhausts all computational resources, if it is applied to practical problem instances.

Instead of using partitions, one may also use *divisions.* $\{S_\rho \mid \rho = 1,\ldots,r\}$ is called division of $S$ if $S = \bigcup_{\rho=1}^{r} S_r$ (no further condition is needed).

There are several simple criteria for stopping further partition somewhere in the partition tree at a set $S_\rho$:

- $S_\rho = \emptyset$
- the optimal solution for $\min\{\boldsymbol{c}^T\boldsymbol{x} \mid \boldsymbol{x} \in S_\rho\}$ is known
- it can be shown that $\min\{\boldsymbol{c}^T\boldsymbol{x} \mid \boldsymbol{x} \in S_\rho\} \geq z_*$, where $z_*$ is the solution value of an element $\boldsymbol{x}_*$, for which we already know $\boldsymbol{x}_* \in S$

Let $R_\rho$ be a set with $R_\rho \supseteq S_\rho$, and let $\hat{z}_* = \min\{\boldsymbol{c}^T\boldsymbol{x} \mid \boldsymbol{x} \in S_\rho\}$, i.e. the optimal value of a relaxation. Let $\hat{\boldsymbol{x}}_*$ be a corresponding value for $\boldsymbol{x}$. Then the following situations allow us to use the criteria from above:

- $R_\rho = \emptyset$
- $\hat{x}_* \in S_\rho$
- $\hat{z}_* \geq z_*$, where $z_*$ is the solution value of a known solution

Algorithms using these partition, relaxation and stopping criteria ideas are frequently called *branch-and-bound* algorithms. A general branch-and-bound algorithm for solving integer programming instances is given by algorithm B.2.

**Algorithm B.2** Branch-and-Bound Algorithm

$z_* := \infty$; $\mathcal{L} = \{S\}$; $l_S := -\infty$
**loop**
  **if** $\mathcal{L} = \emptyset$ **then**
    Stop. If $z_* = \infty$, then the problem is infeasible. Otherwise, $\boldsymbol{x}_*$ is optimal with value $z_*$.
  **end if**
  Choose $S' \in \mathcal{L}$ and a set $R' \supseteq S'$.
  $\mathcal{L} := \mathcal{L} \backslash \{S'\}$
  **if** $R' = \emptyset$ **then**
    **continue**
  **end if**
  $z := \min\{\boldsymbol{c}^T \boldsymbol{x} \mid \boldsymbol{x} \in R'\}$ with optimal solution $\hat{\boldsymbol{x}}$.
  **if** $z \geq z_*$ **then**
    **continue**
  **end if**
  **if** $\hat{\boldsymbol{x}} \in S'$ **then**
    $z_* := z$; $\boldsymbol{x}_* := \hat{\boldsymbol{x}}$
    $\mathcal{L} := \mathcal{L} \backslash \{\hat{S} \mid l_{\hat{S}} \geq z_*\}$
    **continue**
  **end if**
  Choose a partition $\bigcup_{\rho=1}^{r} S_r$ of $S'$.
  $l_{S_\rho} := z$ for each $\rho \in \{1, \ldots, r\}$
  $\mathcal{L} = \mathcal{L} \cup S_1 \cup S_2 \cup \ldots \cup S_r$
**end loop**

In the algorithm, a set $\mathcal{L}$ of sets $S_\rho \subseteq S$ is maintained for which the objective function has to be minimized. Often, $S_\rho$ is identified with the corresponding minimization problem and thus itself is called *problem*. Associated with each $S_\rho \in \mathcal{L}$ is a lower bound $l_{S_\rho}$ such that $\boldsymbol{c}^T \boldsymbol{x} \geq l_{S_\rho}$ for each $\boldsymbol{x} \in S_\rho$. It is possible to have $l_{S_\rho} = -\infty$. The best known solution value that the algorithm has found so far is $z_*$. $z_* = \infty$ means that no solution has been found yet. If $z_* < \infty$, the corresponding solution is given by $\boldsymbol{x}_*$.

Again, the most popular type of relaxation is the LP relaxation. Many commercial computer codes (like CPLEX, which has been used in our experiments) use this type of relaxation. In this case, a solution $\hat{\boldsymbol{x}} \notin S'$ has at least one fractional component, say $t < x_i < t+1$ for an index $i \in \{1, \ldots, n\}$ and an integer $t$. A possible partition then consists of the sets $S_1 := S' \cap \{\boldsymbol{x} \in S \mid x_i \leq t\}$ and $S_2 := S' \cap \{\boldsymbol{x} \in S \mid x_i \geq t+1\}$. This can be expressed as an additional linear inequality for each set and thus the dual simplex algorithm seems to be a promising method for the solution of the problems arising from $S_1$ and $S_2$.

In an LP-based branch-and-bound process, several decisions have to be made. Essentially these decisions are:

- choice of $S' \in \mathcal{L}$

- choice of a fractional component of $\hat{\boldsymbol{x}} \notin R^I$, if it has

The first decision is often called *node selection,* the second decision *choice of the branching variable.* We present some suggestions for these decisions here. For further information we refer to [46], for details concerning availability of such strategies in the commercial software we have used, see [23] and [34].

A widely used node selection rule is the *depth first search* rule. The nodes of the branch-and-bound tree (although a problem set $\mathcal{L}$ is maintained, it can be interpreted as a tree structure) are visited in a depth first search order. Another rule is the *best bound* rule. In this case, the element $S' \in \mathcal{L}$ with

$$l_{S'} = \min_{\hat{S} \in \mathcal{L}} l_{\hat{S}}$$

is selected. By doing this, we try to improve the lower bound on the solution for $S$. Recall that in the branch and bound algorithm, $z_*$ is an upper bound for the solution, $\min_{\hat{S} \in \mathcal{L}} l_{\hat{S}}$ a lower bound.

The choice of the branching variable is frequently done by a *maximum infeasibility* or by a *minimum infeasibility* rule. In the first case the fractional component $x_i$ where $x_i - \lfloor x_i \rfloor$ is "closest to $\frac{1}{2}$" is selected, in the latter case the component which is closest to an integer value. Another rule that has been successfully applied to practical problems is the *strong branching* rule (a description of this is given in [61], for example).

An advantage of branch-and-bound algorithms, compared with relaxation iteration algorithms, is the possible generation of feasible (but not necessarily optimal) solutions while examining some tree nodes. From the lower bounds of all remaining tree nodes and the best known solution, an optimality gap can be calculated (i.e. one knows an interval containing the optimal solution value).

The ideas of *cutting planes* and *branch-and-bound* can be combined effectively:

- *Cut-and-branch*: These algorithms start with cutting planes, until some stopping criterion is fulfilled. Then a branch-and-bound process is started on the problem with the added constraints.
- *Branch-and-cut*: In this case, at every node of the branch-and-bound tree, a cutting plane algorithm is started. As soon as a stopping criterion is fulfilled, the branching is continued and the generated cuts are applied in the complete respective subtree. Note that in such an algorithm, cutting planes only valid for a subtree can be applied.

### Preprocessing

Sometimes the size of a MIP can be reduced before actually starting to solve it. By looking at the specific problem structure, one can often find variables whose values in an optimal solution (or even in a feasible solution) can be easily determined in advance. Thus, they can be replaced by constant values. This process is called *variable fixing.* Similarly, one may detect redundant constraints.

Often, coefficients of the constraint matrix can be modified in such a way that the feasible set remains unchanged, but non-integer extreme points of the corresponding LP are avoided. Such a procedure is called *coefficient reduction* and is especially helpful for cutting plane algorithms (see section B.2). An example is given in figure B.2, where changing the constraint $-\frac{2}{3}x_1 + x_2 \leq 0$ to $-\frac{1}{2}x_1 + x_2 \leq 0$ gives the same feasible set, but leads to an integer optimal solution already for the LP relaxation.

For some problems with a particular structure, coefficient reduction schemes are known.

A more detailed investigation of preprocessing techniques can be found in [35] and [54].

$$
\begin{array}{llrcrcl}
\min & & & - & x_2 & & \\
 & & & & & & \\
\text{s.t.} & - & \frac{2}{3}x_1 & + & x_2 & \leq & 0 \\
 & & x_1 & & & \leq & 2 \\
 & & & - & x_2 & \leq & 0 \\
 & & & & & & \\
 & & x_1, & & x_2 & \in & \mathbb{Z}
\end{array}
$$

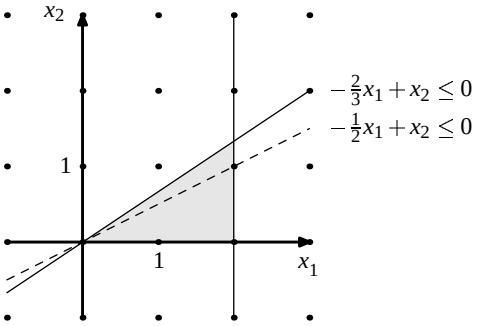



Figure B.2: Changing a coefficient leads to an integer optimal solution for LP here

Let $P = \{\boldsymbol{x} \in \mathbb{R}^n \mid A\boldsymbol{x} \leq \boldsymbol{b}\}$ and $C = \text{conv}(\{\boldsymbol{x} \in \mathbb{Z} \mid A\boldsymbol{x} \leq \boldsymbol{b}\})$. We know that $C \subseteq P$. By coefficient reduction as well as by the addition of cutting planes we try to find polyhedra $P'$ with $C \subseteq P' \subsetneq P$, hoping that a relaxation from $\{\boldsymbol{x} \in \mathbb{Z}^n \mid A\boldsymbol{x} \leq \boldsymbol{b}\}$ to $P'$ gives better bounds for the solution or less non-integral extreme points than a relaxation to $P$.

# Appendix C

# Shortest Path Problems

Let $G = (V,A)$ be a directed graph, and suppose that each arc $a$ is associated with a *length* or *cost* $\mu_a$. Let $|V| =: n$, $|A| =: m$ and let the nodes be denoted by $1,\ldots,n$. The *shortest path problem* is to find a minimum cost path from a source node to a destination node. In order to formalize this idea and to discuss algorithms for solving shortest path problems, we shortly focus on some algebraic structures related to such problems. For details, we refer to [63].

**Definition:** A nonempty set $H$ with internal composition $\oplus : H \times H \to H$ is called *semigroup*, if

$$a \oplus (b \oplus c) = (a \oplus b) \oplus c \qquad \text{for all } a,b,c \in H.$$

A semigroup is called *monoid* if it contains an element $e$ with

$$e \oplus a = a \oplus e = a \qquad \text{for all } a \in H.$$

In this case, $e$ is said to be a neutral element of $H$. If a neutral element exists, it is always uniquely determined. A semigroup is called *commutative*, if

$$a \oplus b = b \oplus a \qquad \text{for all } a,b \in H.$$

**Definition:** A commutative semigroup is called *ordered*, if

$$a \leq b \Rightarrow a \oplus c \leq b \oplus c \qquad \text{for all } a,b,c \in H.$$

An element $a$ of an ordered semigroup is said to be *positive* if

$$b \leq a \oplus b \qquad \text{for all } b \in H.$$

**Definition:** Let $(R,\oplus)$ be a commutative monoid with neutral element 0 and let $(R,\otimes)$ be a (not necessarily commutative) monoid with neutral element 1, where $0 \neq 1$. $(R,\oplus,\otimes)$ is called a *semiring with unity* 1 *and zero* 0, if

$$\left.\begin{array}{rcl} a \otimes (b \oplus c) & = & (a \otimes b) \oplus (a \otimes c) \\ (b \oplus c) \otimes a & = & (b \otimes a) \oplus (c \otimes a) \\ 0 = a \otimes 0 & = & 0 \otimes a \end{array}\right\} \qquad \text{for all } a,b,c \in R.$$

We will shortly speak of the semiring $R$. The first two conditions are called laws of *distributivity*. If all elements of $R$ are idempotent with respect to $\oplus$, $R$ is called *idempotent semiring*. If $(R, \otimes)$ is a commutative monoid, $R$ is called *commutative semiring*.

Let $p := (a_1, \ldots, a_r)$ be a path in the graph $G$ and let the arc cost values be elements of a commutative semiring. The weight $w(p)$ of the path is then defined as

$$w(p) := \bigotimes_{k=1}^{r} \mu_{a_k}.$$

Let $P_{ij}$ denote the set of all paths from $i$ to $j$ in $G$. The problem of determining

$$\mu^*(i,j) := \bigoplus_{p \in P_{ij}} w(p)$$

for a pair of nodes $(i, j)$ and a corresponding path is called *algebraic path problem*. It is common to define $\mu^*(i,i) := 1$. If $R = (\mathbb{R} \cup \{\infty\}, \min, +)$ with zero $\infty$ and unity 0, the algebraic path problem is the classical *shortest path problem*. In this case, it is common to define $\mu^*(i,j) = \infty$ if $(i,j) \notin A$.

**Definition:** A semiring is called *complete* if the following conditions are fulfilled:

- $\bigoplus_{i \in I} a_i \in R$ is well defined for countable sets $I$, $a_i \in R$ for all $i \in I$
- $\bigoplus_{i \in I} a_i = \bigoplus_{j \in J} \left( \bigoplus_{i \in I_j} a_i \right)$ for partitions $(I_j, j \in J)$ of $I$
- $b \otimes \left( \bigoplus_{i \in I} a_i \right) = \bigoplus_{i \in I} (b \otimes a_i)$ and $\left( \bigoplus_{i \in I} a_i \right) \otimes b = \bigoplus_{i \in I} (a_i \otimes b)$ for all $b \in R$

## C.1 Classical Shortest Path Problem

We will now consider the classical shortest path problem with possible negative arc costs in the semiring $(\mathbb{R} \cup \{\infty\}, \min, +)$. If $P_{ij} \neq \emptyset$ and if $G$ does not contain circuits of negative weight, then there exists a shortest path from $i$ to $j$. Shortest path problems (with possibly negative arc costs) can be solved by one of the classical label correcting methods described, for example, in [1]. Those methods either solve the problem or detect a circuit with negative weight in polynomial time (note that in this case there is no solution for the problem).

Note that for shortest path weights the *triangle inequality*

$$\mu^*(i,j) \otimes \mu^*(j,k) \geq \mu^*(i,k) \qquad \text{for all } i,j,k \in N$$

is valid. If one wishes to have all shortest paths from one source node $u$ to all other nodes $v$ of the graph, this can be described by a *shortest path tree*, which is a tree with root $u$ where every uniquely determined path from $u$ to another node $v$ is a shortest path from $u$ to $v$.

### Shortest Path Algorithms

We will briefly describe label setting or label correcting algorithms to calculate shortest paths from a fixed node 1 to all other nodes of a graph. These algorithms are based on iteration schemes and assign a label $\lambda(i)$ to each node $i$. During an iteration, $\lambda(i)$ gives the length of the best path from node 1 to node $i$ found so far. Initially, the labels are defined by $\lambda(1) = 0$ and $\lambda(i) = \infty$ for all nodes $i \neq 1$.

In each iteration of the algorithms, a set of nodes (the so called *candidate list* $L$) is scanned, which means that for each node $i$ in this list, all nodes $j$ with an arc $a : i \to j$ are investigated. If

$$\lambda(j) > \lambda(i) \otimes \mu_a,$$

the path length from node 1 to node $j$ can be improved by combining the path to node $i$ with length $\lambda(i)$ with the arc $a : i \to j$ of length $\mu_a$ (where combining means the use of the semiring operation "$\otimes$"). In this case, $\lambda(j)$ is set to $\lambda(i) \otimes \mu_a$. After an iteration, a new candidate list is obtained by the set of improved nodes. Initially the candidate list only contains the root node 1.

General label correcting methods choose a sublist $L' \subseteq L$ to be scanned. The iteration process stops if $L = \emptyset$.

If there are negative arc costs, the Bellman-Ford algorithm should be used (algorithm C.1, which can be easily adapted in order to calculate not only the shortest path weights, but also the paths themselves), where at each iteration, the complete list $L' := L$ is scanned. If after $n$ iterations the candidate list is not empty, the shortest path problem is not soluble, i.e. the graph contains a circuit of negative length.

**Algorithm C.1** Bellman-Ford Algorithm

$\lambda(1) := \mu_{1j}$ for each $j \in \{1, \ldots n\}$
$L := \{1\}$
**for** $k = 1$ **to** $n$ **do**
  $\hat{L} := \emptyset$
  **for** each $l \in L$ **do**
    **for** each arc $a : l \to j$ **do**
      **if** $\lambda(j) > \lambda(l) \otimes \mu_a$ **then**
        $\lambda(j) := \lambda(l) \otimes \mu_a$
        $\hat{L} := \hat{L} \cup \{j\}$
      **end if**
    **end for**
  **end for**
  $L := \hat{L}$
**end for**
**if** $L = \emptyset$ **then**
  Stop. Shortest path weights from node 1 to all other nodes have been calculated.
**end if**
Stop. The graph contains a circuit with negative weight.

If all arcs have a non-negative cost $\mu_a \geq 0$, the shortest path problem is soluble, and the label setting method proposed by Dijkstra (algorithm C.2, which again can be adapted to determine the corresponding paths) is a very efficient algorithm. During each step, the scan list $L' := \{i_*\} \subseteq L$ contains exactly the node $i_* \in L$ with minimum label $\lambda(i_*) := \min\{\lambda(i) \mid i \in L\}$. The assumption that all arc costs are non-negative guarantees that the shortest path from node 1 to node $i_*$ has already been found, i.e. $\lambda(i_*) = \mu^*(1, i_*)$. Thus, if we are only interested in the shortest path from node 1 to a fixed goal node, we do not have to run the algorithm until $L = \emptyset$, but may stop whenever the goal node has been selected as scan node.

**Algorithm C.2** Dijkstra's Algorithm

$\lambda(j) := \mu_{1j}$ for each $j \in \{1, \ldots, n\}$
$N := V \backslash \{1\}$
**loop**
  Determine $k \in N$ with $\lambda(k) = \min\{\lambda(j) \mid j \in N\}$.
  $N := N \backslash \{k\}$ /* $L' := \{k\}$ */
  **if** $N = \emptyset$ **then**
    Stop. Shortest path weights from node 1 to all other nodes have been calculated
  **end if**
  $\lambda(j) := \min\{\lambda(j), \lambda(k) \otimes \mu_{kj}\}$ for all $j \in N$
**end loop**

## C.2 Gauss-Jordan Method

For a complete semiring, the *generalized Gauss-Jordan method* (algorithm C.3) can be used to determine shortest path weights for all pairs of nodes simultaneously (cf. [63]). For an element $a$ of a semiring, define

$$a^* := 1 \oplus a \oplus (a \otimes a) \oplus (a \otimes a \otimes a) \oplus \ldots$$

**Algorithm C.3** Generalized Gauss-Jordan Method

**for** each $(i, j) \in V \times V$ **do**
  Set $M_{ij} := 1$ if $i = j$ and $M_{ij} := \bigoplus_{a: i \to j} \mu_a$ otherwise
**end for**
**for** $k = 1$ **to** $n$ **do**
  $M_{kk} := M_{kk}^*$
  $M_{ik} := M_{ik} \otimes M_{kk}$ for all $i \neq k$
  $M_{kj} := M_{kk} \otimes M_{kj}$ for all $j \neq k$
  $M_{ij} := M_{ij} \oplus (M_{ik} \otimes M_{kj})$ for all $i, j \neq k$
**end for**
Stop. Shortest path weights are given by $\mu^*(i, j) = M_{ij}$

## C.3 Feasible Differential Problem

Let $G = (V,A)$ be a directed graph, where each arc $a \in A$ is associated with a span $[l_a, u_a]$ (in a non-periodic sense). The *feasible differential problem (FDP)* is to find a potential $\pi$ such that

$$\pi_j - \pi_i \in [l_a, u_a] \qquad \text{for each } a : i \to j.$$

This problem has been examined in [53].

An FDP instance can be solved as a shortest path problem instance on a special graph $G' = (V', A')$, which is constructed in the following way: $V' = V$, $A' = A \cup \{a' : j \to i \mid a : i \to j \in A\}$. If $a : i \to j \in A$, the arc $a' : j \to i$ is called *counter arc* of $a$. Each arc is assigned a length $\mu_a$ with

$$\mu_a := \begin{cases} u_a & \text{if } a \in A \\ -l_a & \text{if } a \text{ is a counter arc.} \end{cases}$$

Now the FDP instance is feasible if and only if there exists a potential $\boldsymbol{\pi}$ for $G'$ with

$$\pi_j - \pi_i \leq \mu_a \qquad \text{for each } a : i \to j \in A'. \tag{C.1}$$

Let a shortest path from an arbitrary node, say node 1, to all other nodes of $G'$ exist (with $\mu_{a_1} \otimes \mu_{a_2} := \mu_{a_1} + \mu_{a_2}$ for $a_1, a_2 \in A'$). In this case, $G'$ does not contain negative circuit. The inequality

$$\mu^*(1,j) \leq \mu^*(1,i) + \mu_a,$$

which is valid for every arc $a : i \to j$ of $G'$ shows that the potential defined by $\pi_i := \mu^*(1,i)$ fulfills inequality (C.1). Conversely, if there is a negative circuit in $G'$, inequality (C.1) cannot be satisfied for the arcs of that circuit.

Consider a circuit with incidence vector $\boldsymbol{\gamma}'$ of $G'$. By traversing each counter arc in negative direction, this circuit can be uniquely described by a *cycle* in $G$. Let $\boldsymbol{\gamma}^+$ ($\boldsymbol{\gamma}^-$) denote the incidence vectors of the set of arcs of this cycle which are traversed in positive (negative) direction. Then the path length of the circuit can be expressed by

$$\boldsymbol{\mu}^T \boldsymbol{\gamma}' = \boldsymbol{u}^T \boldsymbol{\gamma}^+ - \boldsymbol{l}^T \boldsymbol{\gamma}^-.$$

Therefore, the shortest path problem in $G'$ is solvable if and only if for all cycles in $G$ (with incidence vectors $\boldsymbol{\gamma}^+$ and $\boldsymbol{\gamma}^-$ as described above),

$$\boldsymbol{u}^T \boldsymbol{\gamma}^+ - \boldsymbol{l}^T \boldsymbol{\gamma}^- \geq 0. \tag{C.2}$$

When solving PESP instances, the following subproblem plays an important role: Suppose that we have a solution $\boldsymbol{\pi}$ of an FDP instance with spans $[\underline{d}_a, \overline{d}_a]$ for each arc $a$ and exactly one of the bounds for one arc has to be modified. The task now is to find a feasible potential for the new spans or to prove infeasibility.

Assume that the lower bound of $q : u \to v$ was raised from $\underline{d}_q$ to $\underline{d}'_q > \underline{d}_q$. If $\pi_v - \pi_u \geq \underline{d}'_q$, the potential is still feasible for this tightened problem. Now suppose $\pi_v - \pi_u < \underline{d}'_q$. In this case, we have to raise the tension $x_q = \pi_v - \pi_u$ at least for the amount $\delta := \underline{d}'_q - \pi_v + \pi_u$.

Consider again the bi-directed graph $G'$ and define another arc length $\mu'_a$ for each arc $a : i \to j$ by $\mu'_a := \overline{d}_a - \pi_j + \pi_i$ if $a \in A$ and $\mu'_a := -\underline{d}_a + \pi_j - \pi_i$ if $a$ is a counter arc (with $\mu'_{a_1} \otimes \mu'_{a_2} := \mu'_{a_1} + \mu'_{a_2}$). For all arcs $a \in A, a \neq q$ the arc length for $a$ and its counter arc are positive, and therefore the Dijkstra algorithm (algorithm C.2) can be applied to find a shortest path from $u$ to $v$ in $G'$.

Suppose that during the iteration process the goal node $v$ has not yet been selected as scan node, and let the current scan node be $i^*$. If $\lambda(i^*) \geq \delta$, then $\mu'^*(u,v) \geq \mu'^*(u,i) = \lambda(i^*) \geq \delta$. A simple case discussion shows, that then the modified potential

$$\pi'_i := \begin{cases} \pi_i + \lambda(i) - \delta & \text{if } \lambda(i) < \delta \\ \pi_i & \text{otherwise} \end{cases}$$

is a feasible potential for the modified FDP instance. If node $v$ is labeled with $\lambda(v) < \delta$, then there exists a negative circuit for weight $\mu$ in for $G'$, and the modified FDP instance is infeasible: Consider the circuit consisting of the shortest path $(a_1, \ldots, a_r)$ for weight $\mu'$ from node $u$ to node $v$ and the counter arc for $q$. We know that

$$\begin{aligned} \sum_{\substack{k=1 \\ a_k : i \to j \in A}}^{r} \mu'_{a_k} + \sum_{\substack{k=1 \\ a_k : i \to j \text{ counter arc}}}^{r} \mu'_{a_k} &< \delta \\ \sum_{\substack{k=1 \\ a_k : i \to j \in A}}^{r} \overline{d}_{a_k} - \pi_j + \pi_i + \sum_{\substack{k=1 \\ a_k \text{ counter arc to } a : i \to j}}^{r} -\underline{d}_a + \pi_j - \pi_i &< \underline{d}_q - \pi_v + \pi_u \\ \pi_u - \pi_v + \sum_{\substack{k=1 \\ a_k : i \to j \in A}}^{r} \overline{d}_{a_k} + \sum_{\substack{k=1 \\ a_k \text{ counter arc to } a}}^{r} -\underline{d}_a &< \underline{d}_q - \pi_v + \pi_u, \end{aligned}$$

which is a contradiction to (C.2) In this case, the arcs of the circuit (or the corresponding cycle arcs in $G$) are called *blocking arcs*.

The modified Dijkstra algorithm (which will stop as soon as a negative circuit has been found or $\lambda(i^*) \geq \delta$ for a node $i^*$) will be denoted by $\mathrm{Dij}^{\mathrm{lower}}(\delta, \underline{\boldsymbol{d}}, \overline{\boldsymbol{d}}, \boldsymbol{\pi}, \boldsymbol{\gamma})$ for the tension lowering version and by $\mathrm{Dij}^{\mathrm{raise}}(\delta, \underline{\boldsymbol{d}}, \overline{\boldsymbol{d}}, \boldsymbol{\pi}, \boldsymbol{\gamma})$ for the tension raising version.

# List of Algorithms

# List of Symbols

| | | |
|---|---|---|
| NP | Complexity class | 102 |
| P | Complexity class | 101 |
| | | |
| $A_{\mathcal{G}}$ | Arc set of an event graph | 14 |
| $C^{\text{fix}}$ | Fixed cost per motor unit | 22 |
| $C^{\text{fixC}}$ | Fixed cost per coach | 22 |
| $C^{\text{km}}$ | Cost per km per motor unit | 22 |
| $C^{\text{kmC}}$ | Cost per km per coach | 22 |
| $C_{\tau}^{\text{fix}}$ | Fixed cost per motor unit of type $\tau$ | 24 |
| $C_{\tau}^{\text{fixC}}$ | Fixed cost per coach of type $\tau$ | 24 |
| $C_{\tau}^{\text{km}}$ | Cost per km per motor unit of type $\tau$ | 24 |
| $C_{\tau}^{\text{kmC}}$ | Cost per km per coach of type $\tau$ | 24 |
| $E$ | Set of edges | 9 |
| $G$ | Graph | 9 |
| $N_e$ | Number of travelers (passenger demand) on edge $e$ | 22 |
| $Per$ | Periodic sets | 36 |
| $T$ | Basic time period | 9 |
| $V$ | Set of nodes | 9 |
| $V_{\mathcal{G}}$ | Node set of an event graph | 14 |
| $\underline{W}$ | Minimal number of coaches per train | 22 |
| $\overline{W}$ | Maximal number of coaches per train | 22 |
| $\underline{W}_{\tau}$ | Minimal number of coaches per train of type $\tau$ | 24 |
| $\overline{W}_{\tau}$ | Maximal number of coaches per train of type $\tau$ | 24 |
| | | |
| $\mathcal{C}$ | Set of periodic interval constraints | 13 |
| $\mathcal{E}$ | Set of periodic events | 10 |
| $\mathcal{E}^0$ | Set of corresponding individual events with index 0 for $\mathcal{E}$ | 10 |
| $\hat{\mathcal{E}}$ | Set of individual events | 10 |
| $\mathcal{F}_r$ | Set of possible frequencies for line $r$ | 20 |
| $\mathcal{G}$ | Event graph of a PESP instance | 14 |

| | | |
|---|---|---|
| $\mathcal{J}$ | Set of joining constraints | 18 |
| $\mathcal{Q}$ | Unbounded timetable polyhedron | 39, 52 |
| $\mathcal{Q}_{\mathcal{T}}$ | Bounded timetable polyhedron | 52 |
| $\mathcal{R}$ | Set of lines | 9 |
| $\hat{\mathcal{R}}$ | Set of lines causing infeasibility of an FSP instance | 67 |
| $\mathcal{R}^P$ | Set of possible lines | 20 |
| $\mathcal{T}$ | Set of train types | 24 |
| | or: Spanning tree for a PESP instance | 34 |
| $\mathcal{T}_r$ | Set of feasible train types for line $r$ | 24 |
| $\mathcal{X}$ | Set of train change time constraints | 91 |
| $\mathcal{Z}$ | Set of feasible modulo parameters | 39, 52 |
| $\mathcal{Z}_{\mathcal{T}}$ | Set of feasible modulo parameters with zero on a spanning tree | 41, 52 |
| | | |
| $\mathfrak{C}$ | Capacity of one coach | 22 |
| $\mathfrak{C}_\tau$ | Capacity of one coach of train type $\tau$ | 24 |
| $\mathfrak{N}_{\mathcal{G}}$ | Node arc incidence matrix for PESP event graph $\mathcal{G}$ | 34 |
| | | |
| $a^v_{r,\mu}$ | Arrival of line $r$, direction $\mu$, at node $v$ | 10 |
| $d_r$ | Length of circulation of line $r$ | 20 |
| $d^v_{r,\mu}$ | Departure of line $r$, direction $\mu$, at node $v$ | 10 |
| $\underline{lfr}_e$ | Minimal line frequency on edge $e$ | 22 |
| $\overline{lfr}_e$ | Maximal line frequency on edge $e$ | 22 |
| $t_r$ | Circulation time for a train of line $r$ | 20 |
| $\hat{t}_r$ | Estimated circulation time for a train of line $r$ | 20 |
| $\hat{t}_{r,\tau}$ | Estimated circulation time for line $r$ with train type $\tau$ | 24 |
| $\underline{trav}^{vv'}_\tau$ | Minimum travel time for trains of type $\tau$ from $v$ to $v'$ | 24 |
| $\overline{trav}^{vv'}_\tau$ | Maximum travel time for trains of type $\tau$ from $v$ to $v'$ | 24 |
| $\underline{turn}$ | Minimum turnaround time | 24 |
| $\overline{turn}$ | Maximum turnaround time | 24 |
| $w_r$ | Number of coaches of trains of line $r$ | 21 |
| $w_{r,f}$ | Number of coaches of trains of frequency $f$ for line $r$ | 21 |
| $w_{r,\tau}$ | Number of coaches of type $\tau$ for trains of line $r$ | 24 |
| $w_{r,f,c}$ | Line $r$ is used with frequency $f$ and $c$ coaches (indicator variable) | 23 |
| $\underline{wait}$ | Minimum waiting time | 24 |
| $\overline{wait}$ | Maximum waiting time | 24 |
| $x_r$ | Frequency of line $r$ | 21 |
| $x_{r,f}$ | Line $r$ is used with frequency $f$ (indicator variable) | 21 |
| $x_{r,\tau}$ | Line $r$ is used with train type $\tau$ | 24 |

| | | |
|---|---|---|
| $\Gamma$ | Network matrix of a graph | 34 |
| $\Pi(\boldsymbol{z})$ | Set of feasible potentials for fixed modulo parameters | 39 |
| | | |
| $\hat{\gamma}_r$ | Estimated number of train compositions required for line $r$ | 20 |
| $\boldsymbol{\theta}_k$ | Column belonging to node $k$ in the transposed incidence matrix | 34 |
| $\mu^*(i,j)$ | Shortest path weight from node $i$ to node $j$ | 112 |
| $\pi$ | Schedule for periodic events | 10 |
| $\hat{\pi}$ | Schedule for individual events | 9 |
| | | |
| $[a,b]_T$ | Periodic extension of the interval $[a,b]$ with period $T$ | 10, 34 |
| $\boldsymbol{p}^+, \boldsymbol{p}^-$ | Positive and negative part of a vector | 33 |
| $a \equiv b \bmod T$ | Modulo operation | 14 |
| $(\cdot) \bmod t$ | Modulo projection | 35 |
| $a : i \to j$ | $a$ is an arc from $i$ to $j$ | 33 |
| $\mathrm{Dij}^{\mathrm{lower}}$, $\mathrm{Dij}^{\mathrm{raise}}$ | Modified Dijkstra procedures | 116 |
| $O(f(n))$ | Complexity class | 101 |

# Bibliography


[1] R.K. Ahuja, T.L. Magnanti and J.B. Orlin. *Network Flows — Theory, Algorithms, and Applications.* Prentice Hall, 1993.

[2] A.A. Assad. Models for Rail Transportation. *Transportation Research Part A* 14, 205–220, 1980.

[3] P. Baron. Transportation in Germany: A Historical Overview. *Transportation Research Part A* 29, 9–20, 1995.

[4] J.H.A. Van den Berg and M.A. Odijk. DONS: Computer Aided Design of Regular Service Timetables. In *Proceedings of CompRail94*, Madrid, 1994.

[5] R.E. Bixby. Private communication, 1999.

[6] U. Brännlund, P.O. Lindberg, A. Nõu, J.-E. Nilsson. Railway Timetabling using Lagrangian Relaxation. *Transportation Science* 32 (4), 358–369, 1998.

[7] H. Braker. *Algorithms and Applications in Timed Discrete Event Systems.* PhD thesis, University Delft. 1993.

[8] M.R. Bussieck, P. Kreuzer and U.T. Zimmermann. Optimal lines for railway systems. *European J. Oper. Res.*, 96, 54–63, 1996.

[9] M.R. Bussieck, T. Winter and U.T. Zimmermann. Discrete optimization in public rail transport. *Math. Programming*, 79 (1–3), 415–444, 1997.

[10] M.R. Bussieck. Optimal Lines in Public Rail Transport. PhD thesis, Technische Universität Braunschweig, 1998.

[11] A. Caprara, M. Fischetti, P. Toth, D. Vigo and P.L. Guida. Algorithms for railway crew management. *Math. Programming*, 79 (1–3), 125–141, 1997.

[12] M. Carey. A model and strategy for train pathing with choice of lines, platforms, and routes. *Transportation Research B* 28 (5), 333–353, 1994.

[13] M. Carey. Extending a train pathing model from one-way to two-way track. *Transportation Research B* 28 (5), 395–400, 1994.

[14] T. Christof. Ein Verfahren zur Transformation zwischen Polyederdarstellungen. Master's thesis. Universität Augsburg, 1991.

[15] V. Chvátal. *Linear programming*. Freeman and Company, 1983.

[16] M.T. Claessens. De kost-lijnvoering. Master's thesis, University of Amsterdam, 1994.

[17] M.T. Claessens, N.M. van Dijk and P.J. Zwaneveld. Cost optimal allocation of rail passenger lines. *European J. Oper. Res.*, 110 (3) 474–489, 1998.

[18] J.-F. Cordeau, P. Toth and D. Vigo. A Survey of Optimization Models for Train Routing and Scheduling. *Transportation Science*, 32 (4), 380–404, 1998.

[19] J.R. Daduna and S. Voß. Practical Experiences in Schedule Synchronization. *Computer-Aided Transit Scheduling: Proceedings of the 6th International Workshop on Computer-Aided Scheduling of Public Transport,* Lecture Notes in Economics and Mathematical Systems, volume 430, 1995.

[20] Deutsche Bahn. Geschäftsbericht 1998.

[21] Deutsche Bahn. Umweltbericht 1998.

[22] W. Filz. Erste Fahrplankonferenz der deutschen Eisenbahnen (20.4.1871). *Zeitzeichen* 35-960420 (manuscript for radio transmission), Westdeutscher Rundfunk, 1996.

[23] GAMS Development Corp. *GAMS — A User's Guide*, 1998.

[24] GAMS Development Corp. *GAMS — The Solver Manuals*, 1999.

[25] M.R. Garey and D.S. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness.* Freeman and Company, 1979.

[26] M. Grötschel, A. Löbel and M. Völker. *Optimierung des Fahrzeugumlaufs im öffentlichen Nahverkehr*. In K.-H. Hoffmann, W. Jäger, T. Lohmann and H. Schnuck (editors), *Mathematik – Schlüsseltechnologie für die Zukunft*, Springer, 1997.

[27] R. Göbertshahn. Der integrale Taktfahrplan – Fundament der neuen Nahverkehrsstrategie von Deutscher Bundesbahn und Deutscher Reichsbahn. *Die Deutsche Bahn* 5/93, 357–362, 1993.

[28] R.E. Gomory. Outline of an algorithm for integer solutions of linear programs. *Bull. Amer. Math. Soc.*, 64, 275–278, 1958.

[29] M. Grötschel, L. Lovasz and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer, Berlin, 1988.

[30] I. Gertsbakh and P. Serafini. Periodic Transportation Schedules with Flexible Departure Times. *European J. Oper. Res.*, 50, 298–309, 1991.

[31] P. Heusch, F. Meisgen and E. Speckenmeyer. CATS — Computer Aided Tram Scheduling. Report No. 97-262, Universität zu Köln, 1997.

[32] A. Higgins, E. Kozan and L. Ferreira. Optimal Scheduling of Trains on a Single Line Track. *Transportation Research B*, 30 (2), 147–161, 1996.

[33] J.S. Hooghiemstra. Design of regular interval timetables for strategic and tactical railway planning. In J. Allan, C.A. Brebbia, R.J. Hill, G. Sciutto, and S. Sone, editors. *Railway Systems and Management*, volume 1 of *Computers in Railways V*. Computational Mechanics Publications, 1996.

[34] ILOG. *ILOG CPLEX 6.5 Reference Manual.* 1999.

[35] E.L. Johnson, M.M. Kostreva and U.H. Suhl. Solving 0-1-Integer Programming Problems Arising from Large Scale Planning Models. *Oper. Res.*, 33, 803-819, 1985.

[36] M. Kolonko, K. Nachtigall and S. Voget. Exponat der Universität Hildesheim auf der CeBit 96: Optimierung von Taktfahrplänen mit genetischen Algorithmen. *Hildesheimer Informatik-Berichte*, 8/96, 1996.

[37] M. Krista. Verfahren zur Fahrplanoptimierung dargestellt am Beispiel der Synchronzeiten. PhD thesis, Technische Universität Braunschweig, 1996.

[38] L.G. Kroon and P.J. Zwaneveld. Routing trains through railway stations including shunting decisions. In: P. Kleinschmidt, A. Bachem, U. Derigs, D. Fischer, U. Leopold-Wildburger and R. Möhring, editors: *Operations Research Proceedings 1995*, 438–444, Springer, Berlin, 1995.

[39] A. Löbel. *Optimal vehicle scheduling in public transit.* PhD thesis, TU Berlin, 1997.

[40] M.J. Maher. Inferences on trip matrices from observations on link volumes: A Bayesian statistical approach. *Transportation Research B*, 17, 435–447, 1983.

[41] K. Nachtigall. A Branch and Cut Approach for Periodic Network Programming. *Hildesheimer Informatik-Berichte*, 29/94, 1994.

[42] K. Nachtigall. Cutting Planes for a Polyhedron Associated with a Periodic Network. Technical Report IB 112-96/17, DLR, Braunschweig, 1996.

[43] K. Nachtigall. Has PESP an Error? 1997.

[44] K. Nachtigall and S. Voget. Minimizing Waiting Times in Integrated Fixed Interval Timetables by upgrading Railway Tracks. *European J. Oper. Res.*, 103, 610–627, 1997.

[45] K. Nachtigall. Periodic Network Optimization and Fixed Interval Timetables. Habilitationsschrift. Universität Hildesheim, 1998.

[46] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, 1988.

[47] M.A. Odijk. Construction of Periodic Timetables — Part I: A Cutting Plane Algorithm. Technical Report 94-61, Department of Mathematics and Computer Science. Delft University of Technology, 1994.

[48] M.A. Odijk. Construction of Periodic Timetables — Part II: An Application. Technical Report 94-71, Department of Mathematics and Computer Science. Delft University of Technology, 1994.

[49] M.A. Odijk. A constraint generation algorithm for the construction of periodic railway timetables. *Transportation Research B*, 30 (6), 1996.

[50] C. Oltrogge. *Linienplanung für mehrstufige Bedienungssysteme im öffentlichen Personenverkehr*. PhD thesis, Technische Universität Braunschweig, 1994.

[51] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization — Algorithms and Complexity*. Prentice Hall, 1982.

[52] C. Pöppe. Fahrplan-Algebra. *Spektrum der Wissenschaft* 11, 18–21, 1995.

[53] R.T. Rockafellar. *Network Flows and Monotropic Optimization*. John Wiley & Sons, 1984.

[54] M.W.P. Savelsbergh. Preprocessing and probing techniques for mixed integer programming problems. *ORSA J. Comput.*, 6 (4), 445–454, 1994.

[55] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1986.

[56] A. Schrijver and A. Steenbeek. Dienstregelingontwikkeling voor Railned. Technical Report, Centrum voor Wiskunde en Informatica, 1994.

[57] A. Schrijver. Routing and Timetabling by Topological Search. *Proceedings International Congress of Mathematicians*, Berlin, 1998.

[58] H.D. Sherali, R. Sivanandan, A.G. Hobeika. A linear programming approach for synthesizing origin-destination trip tables from link traffic volumes. *Transportation Research B*, 28, 213–233, 1994.

[59] P. Serafini and W. Ukovich. A mathematical model for periodic scheduling problems. *SIAM J. Disc. Math.* 2 (4), 1989.

[60] P. Serafini and W. Ukovich. A mathematical model for the fixed-time traffic control problem. *European J. Oper. Res.* 42, 152–165, 1989.

[61] S. Thienel. *ABACUS — A Branch-And-CUt System*. PhD Thesis, Universität Köln.

[62] M. Voorhoeve. Rail Scheduling with Discrete Sets. Unpublished report, Eindhoven University of Technology, The Netherlands, 1993.

[63] U.T. Zimmermann. *Linear and Combinatorial Optimization in Ordered Algebraic Structures*. Number 10 in Annals of Discrete Mathematics, 1981.

# Index

# Deutsche Zusammenfassung

Die vorliegende Arbeit befaßt sich mit Fahrplanoptimierung unter Berücksichtigung der Verhältnisse beim spurgeführten, öffentlichen Personenverkehr. Insbesondere wird davon ausgegangen, daß der Fahrplan sich nach einer bestimmten Zeitperiode (z.B. eine Stunde) wiederholen soll.

Ein Fahrplan besteht aus den Ankunfts- und Abfahrtzeiten der einzelnen Verkehrslinien an bestimmten Punkten im Verkehrsnetz, etwa den Bahnhöfen beim Eisenbahn-Fernverkehr. Fahrpläne lassen sich nach unterschiedlichen Kriterien bewerten. Im Mittelpunkt dieser Arbeit steht die Minimierung der durch einen Fahrplan entstehenden Betriebskosten für die Fahrzeuge.

In Kapitel 1 wird die Fahrplanerstellung als Teil der Verkehrsplanung dargestellt. Diese Planung wird normalerweise als hierarchischer Prozeß betrachtet. Die einzelnen Teilaufgaben wie etwa Linienplanung, Fahrplanung oder Personaleinsatzplanung, werden in dem Kapitel vorgestellt, und es wird aufgezeigt, wie sie sich gegenseitig beeinflussen.

Kapitel 2 stellt mathematische Modelle zur Fahrplanerstellung vor. Eine zentrale Bedeutung innerhalb dieser Arbeit kommt dabei dem sogenannten *Periodic Event Scheduling Problem (PESP)* zu, das im Jahr 1989 von Serafini und Ukovich eingeführt wurde. Das PESP ist ein Zulässigkeitsproblem, berücksichtigt also keine Optimierungsaspekte. Weiterhin werden in dem Kapitel aus der Literatur bekannte Fahrplanbewertungsansätze erläutert. Ein neues Modell zur *kostenoptimalen* Fahrplangestaltung, das sogenannte *Minimum Cost Scheduling Problem (MCSP)*, wird entwickelt. Es kombiniert Ideen des PESPs mit einem von Claessens im Jahr 1994 vorgeschlagenen Kostenkonzept zur Linienoptimierung. Das MCSP läßt sich als gemischt-ganzzahliges lineares Programm darstellen. Darüber hinaus enthält Kapitel 2 Ergebnisse zur Komplexität des PESPs und des MCSPs.

Das PESP wird in Kapitel 3 genauer untersucht. Es werden aus der Literatur bekannte Lösungsalgorithmen vorgestellt. Durch einige Modifikationen an den Verfahren läßt sich die Lösungszeit für aus Praxissicht relevante Probleminstanzgrößen deutlich verkürzen. Desweiteren enthält das Kapitel neue Resultate in Bezug auf die polyedrische Struktur des PESPs. Mit Hilfe dieser Ergebnisse wird ein neues Branch-and-Cut-Verfahren zur Bearbeitung von PESP-Instanzen entwickelt, das noch einmal eine wesentliche Beschleunigung des Lösungsvorgangs ermöglicht.

Eine direkte Lösung der gemischt-ganzzahligen linearen Programme für interessante Verkehrsnetzgrößen mittels kommerzieller Software erwies sich aufgrund zu langer Rechenzeiten und zu hohem Speicherbedarf – selbst bei massivem Hardwareeinsatz – als nicht möglich. In Kapitel 4 wird eine Dekompositionsidee beschrieben und sowohl in ein Schnittebenenverfahren als auch in ein Branch-and-Bound-Verfahren integriert. Als Teilprobleme treten in jeder Iteration bzw. in jedem Knoten PESP-ähnliche Probleme auf. Mit den Verfahren aus diesem Kapitel können in akzeptabler Zeit

Lösungen von hoher, beweisbarer Qualität generiert werden. Für kleinere Verkehrsnetze ist sogar eine exakte Optimierung möglich. Das Kapitel endet mit der Betrachtung eines nichtlinearen gemischt-ganzzahligen Modells, das die Fahrplankosten noch etwas genauer berechnet. Für dieses Modell wird ein exakter Lösungsalgorithmus angegeben, der allerdings für praktische Problemgrößen zu langsam ist.

In Kapitel 5 werden Rechenergebnisse für die in dieser Arbeit vorgestellten neuen Verfahren präsentiert. Dazu wurden von den Eisenbahnbetreibern *Deutsche Bahn AG* und *Nederlandse Spoorwegen* Praxisdaten zur Verfügung gestellt.

Das letzte Kapitel der Arbeit enthält Anregungen für die mathematische Bearbeitung sehr großer Probleminstanzen. Weiterhin wird ein Ausblick auf zukünftige Modelle und Methoden zur Fahrplanoptimierung gegeben.

# Lebenslauf

| | | |
|---|---|---|
| Name: | | Thomas Lindner |
| Geburtsdatum: | | 14. Oktober 1972 |
| Geburtsort: | | Wolfenbüttel |
| Familienstand: | | verheiratet |
| Staatsangehörigkeit: | | deutsch |
| Bildungsgang: | 1979–1992 | Besuch der Grundschule, Orientierungsstufe und des Gymnasiums |
| | 1992–1993 | Zivildienst |
| | 1993–1997 | Studium der Mathematik (Dipl.) an der TU Braunschweig |
| Beschäftigungszeiten: | 1995–1997 | studentische Hilfskraft in der Abteilung Mathematische Optimierung der TU Braunschweig |
| | 1997–2000 | wissenschaftlicher Mitarbeiter in der Abteilung Mathematische Optimierung der TU Braunschweig |